

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Methods and tools for characterizing transient phase-amplitude coupling in neural oscillations

Permalink

<https://escholarship.org/uc/item/1n3689vx>

Author

Martinez Cancino, Ramon

Publication Date

2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

**Methods and tools for characterizing transient phase-amplitude coupling in
neural oscillations**

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Electrical Engineering
(Intelligent Systems, Robotics, and Control)

by

Ramon Martinez Cancino

Committee in charge:

Kenneth Kreutz-Delgado, Chair
Vikash Gilja
Tzyy-Ping Jung
Scott Makeig
Piya Pal
Virginia de Sa

2020

Copyright

Ramon Martinez Cancino, 2020

All rights reserved.

The Dissertation of Ramon Martinez Cancino is approved and is acceptable
in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2020

DEDICATION

Fur Johi und unseren Kleinen Teo Maximilian.

A mis viejos en Cuba.

TABLE OF CONTENTS

Signature Page	iii
Dedication	iv
Table of Contents	v
List of Figures	viii
List of Tables	xi
List of Listings	xii
Acknowledgements	xiii
Vita	xvi
Abstract of the Dissertation	xx
Chapter 1 Introduction	1
1.1 The human brain	2
1.2 Looking into the brain through its electrical activity	5
1.3 Information theory in neurosciences	9
1.4 Goal and specific aims	10
1.5 Chapters overview	11
Chapter 2 Measuring Transient Phase-Amplitude Coupling Using Local Mutual Information	14
2.1 Abstract	14
2.2 Introduction	15
2.3 Methods	19
2.3.1 Ethics statement	19
2.3.2 Traditional PAC measures	22
2.3.3 Information-theoretic measures	25
2.3.4 Defining local mutual information	27
2.3.5 Estimating local mutual information	29
2.3.6 Estimating PAC using local mutual information	32
2.3.7 Surrogate analysis to compute MIPAC significance	35
2.4 Results	36
2.4.1 Simulation results	36
2.4.2 Applying MIPAC to actual ECoG data	47
2.5 Discussion	57
2.5.1 MIPAC estimates of simulated data	57
2.5.2 MIPAC estimates of experimental data	59

2.5.3	Limitations, caveats and future directions	61
2.6	Conclusions	64
2.7	Acknowledgements	64
Chapter 3	What Can Transfer Entropy Tell Us About Phase-Amplitude Coupling in Electrophysiological Signals?	66
3.1	Abstract	66
3.2	Introduction	67
3.3	Information theory and transfer entropy	69
3.4	Approaching PAC estimation with information theory local measures	76
3.5	Simulation results	78
3.6	Estimating PAC with transfer entropy from actual ECoG data	84
3.7	Discussion	88
3.7.1	Computing TE and MIPAC in simulated PAC data	88
3.7.2	Estimating PAC with local TE in actual ECoG data	90
3.7.3	The problem of directional causality in PAC	92
3.7.4	MI and TE, two faces of the PAC process?	93
3.7.5	General considerations for approaching PAC estimation using local transfer entropy	94
3.8	Conclusions	98
3.9	Acknowledgements	99
Chapter 4	Computing Phase Amplitude Coupling in EEGLAB: PACTools	100
4.1	Abstract	100
4.2	Introduction	101
4.3	Introducing the EEGLAB plug-in <i>PACTools</i>	103
4.3.1	Implementation, architecture, and dependencies	103
4.3.2	Methods implemented in <i>PACTools</i>	105
4.3.3	<i>PACTools</i> output format	106
4.3.4	Computing PAC using: <i>pop_pac</i>	107
4.3.5	Visualizing PAC results	109
4.3.6	Code parallelization and high-performance computing	112
4.4	Proof of concept application: Computing and visualizing PAC on simulated data	112
4.4.1	Computing and visualizing PAC on simulated data	112
4.4.2	Computing and visualizing PAC on actual EEG data	114
4.5	Discussion and conclusions	116
4.6	Acknowledgments	118
Chapter 5	The Open EEGLAB Portal Interface: High-Performance Computing with EEGLAB	119
5.1	Abstract	119
5.2	Introduction	120
5.3	The Open EEGLAB Portal	125

5.3.1	NSG and Open EEGLAB Portal structure and capabilities	125
5.3.2	Registering EEGLAB as an NSG application	127
5.3.3	User interaction with the OEP	127
5.4	The EEGLAB <i>nsgportal</i> plug-in	129
5.4.1	Accessing NSG through the plug-in <i>nsgportal</i>	129
5.4.2	Implementation, architecture, and dependencies	129
5.4.3	Preparing <i>nsgportal</i> for use	132
5.4.4	The main plug-in GUI: <i>pop_nsg</i>	133
5.4.5	<i>Nsgportal</i> Matlab-interface command line tools	135
5.4.6	<i>Nsgportal</i> documentation	137
5.5	Applications and examples	138
5.5.1	The test data and simple job example	138
5.5.2	Submitting, managing and retrieving the job using the <i>nsgportal</i> GUI	140
5.5.3	Submitting, managing and retrieving a job using <i>nsgportal</i> command line tools.	143
5.5.4	Accelerating EEGLAB plug-in performance using <i>nsgportal</i> tools .	146
5.6	Discussion	156
5.7	Conclusions	159
5.8	Acknowledgements	160
Chapter 6	Conclusion	161
Appendix A	Phase-amplitude coupled signal modeling	164
A.1	Phase-amplitude coupled signal modeling	164
Appendix B	NSG REST API commands in <i>nsgportal</i>	166
B.1	Submitting a job	167
B.2	Checking job status	167
B.3	Retrieving job results	168
B.4	Deleting a job	168
Appendix C	NSG jobs status list	169
Appendix D	Wakeman-Henson data processing	170
Appendix E	<i>Nsgportal</i> GUI elements	172
Bibliography		176

LIST OF FIGURES

Figure 1.1.	Plates vi & vii of the Edwin Smith Papyrus at the Rare Book Room.	2
Figure 1.2.	Structure of a neuron.	3
Figure 1.3.	Brain's gray and white matter	4
Figure 1.4.	High-density EEG setup.....	6
Figure 2.1.	Schematic of Z -space showing the $k = 2$ nearest-neighbors search for data point z_i	31
Figure 2.2.	MIPAC from a simulated PAC signal with on/off boxcar coupling. . .	38
Figure 2.3.	MIPAC estimates of simulated PAC signals with linearly increasing coupling.	40
Figure 2.4.	MIPAC time course estimate for a simulated PAC signal with absolute value of a sinusoid used as coupling.	41
Figure 2.5.	MIPAC estimation using <i>Instantaneous MIPAC</i> on simulated PAC signals with noise added.....	42
Figure 2.6.	Convergence in MIPAC estimation.....	43
Figure 2.7.	Relationship between MIPAC estimates and signal parameters.....	44
Figure 2.8.	Estimating MIPAC in a simulated noisy multi-trial PAC data set. . .	45
Figure 2.9.	Comparing results of MImi with those of other PAC indices.	46
Figure 2.10.	Electrode locations and MImi comodulograms computed for face and house stimulus-locked trials	49
Figure 2.11.	ERP-images for Channel 16 trials time locked to presentations of face and house image stimuli.	50
Figure 2.12.	Event-related spectral perturbations (ERSPs) and inter-trial coherences (ITCs) for data from channel 16.....	52
Figure 2.13.	Two single face image presentation trials displaying high-frequency activity.	53
Figure 2.14.	Trial-mean MIPAC and ERPAC computed for Channel 16 at f_{phase} 16 Hz and f_{amp} 95 Hz	54

Figure 2.15.	MIPAC-images showing event-related MIPAC time series for individual face image trials.	56
Figure 3.1.	Comodulogram for the simulated PAC signal computed by the time-averaged mutual information phase-amplitude coupling (MIPAC). ..	79
Figure 3.2.	Simulated PAC signal and MIPAC computed between the phase at 6 Hz and amplitude frequencies from 40 to 120 Hz (10-Hz steps)	80
Figure 3.3.	Active information storage as a function of the embedding history k . .	81
Figure 3.4.	Transfer entropy computed between the phase frequency component at 6 Hz and amplitude frequency component at 70 Hz as a function of K-nearest neighbors.	82
Figure 3.5.	Simulated PAC signal and local transfer entropy computed from the phase to the amplitude time series.	83
Figure 3.6.	Delay estimation using transfer entropy.	84
Figure 3.7.	Time-frequency features of ECoG data clip.	85
Figure 3.8.	Time course of ECoG signal and MIPAC computed between the phase at 4-8Hz and amplitude in the band 30-250Hz.	86
Figure 3.9.	Transfer entropy in ECoG data as a function of K-nearest neighbor values.	87
Figure 3.10.	Time courses of ECoG data and of local transfer entropy from the phase to the amplitude time series.	88
Figure 4.1.	<i>PACTools</i> graphical user interface and its components.	108
Figure 4.2.	Graphical user interface for function <i>pop_plotpac</i>	111
Figure 4.3.	Visualizing PAC results using <i>pop_plotpac</i>	113
Figure 4.4.	Assessing PAC on selected independent components of ICA-decomposed EEG data.	116
Figure 5.1.	Citations of the EEGLAB reference paper vs. other open-source EEG analysis software reference paper citations as per the Web of Science (https://www.webofknowledge.com , mid-December 2019).	123
Figure 5.2.	High-level overview of the Open EEGLAB Portal (OEP) workflow. .	126

Figure 5.3.	Graphical user interface of <i>pop_nsginfo</i>	127
Figure 5.4.	Primary point of user interaction with NSG within EEGLAB: The <i>pop_nsg.m</i> graphical user interface window.	134
Figure 5.5.	Workflow of the example plug-in RUNICA_NSg.....	148
Figure 5.6.	Original RELICA workflow.	150
Figure 5.7.	The NSG-capable RELICA plug-in.	152
Figure 5.8.	RELICA processing time on NSG versus a laptop.	154
Figure 5.9.	Cluster exemplar scalp map correlations for best-matching IC pairs in the output of the same RELICA decomposition run (twice) on a laptop and (once) on XSEDE network cluster Comet via the Neuroscience Gateway using <i>nsgportal</i>	155

LIST OF TABLES

Table 2.1.	Example of studies on Phase-Amplitude Coupling.	17
Table 4.1.	Dimensions of the PAC results generated by the methods implemented in <i>PACTools</i>	106

LIST OF LISTINGS

Listing 4.1.	<i>PACTools</i> structure to store results and settings used in the PAC computation.	106
Listing 5.1.	Generic command line call to <i>pop_nsginfo</i> to store NSG user credential information	133
Listing 5.3.	Example command line call running an NSG job from the MATLAB command line using <i>pop_nsg</i>	136
Listing 5.2.	Generic key-value input pairs used to invoke <i>pop_nsg</i>	136
Listing 5.4.	MATLAB command line call to <i>nsg_recurspoll</i>	137
Listing 5.5.	Job script ‘ <i>runica_on_nsg.m</i> ’	139
Listing 5.6.	MATLAB command window output after retrieving job results using the ‘Download job results’ button in the <i>pop_nsg</i> GUI.	142
Listing 5.7.	Using <i>pop_nsg</i> for submitting the <i>runica_on_nsg.zip</i> job for processing at NSG	143
Listing 5.8.	Command line call initiating recursive polling of the job status of job ‘ <i>currentjob</i> ’	144
Listing 5.9.	Command line call to <i>pop_nsg</i> to retrieve results of job ‘ <i>currentjob</i> ’.	145
Listing 5.10.	Command line call to delete the job whose job structure is <i>currentjob</i>	145
Listing B.1.	Code sample B.1	166
Listing B.2.	Code sample B.2	167
Listing B.3.	Code sample B.3	168
Listing B.4.	Code sample B.4	168

ACKNOWLEDGEMENTS

I want to start thanking my family (the Martinez and Cancino from Cuba, the Forte-Cruz from Miami and the Wagner from beautiful Breitbrunn in Bavaria), which has always been my drive, motor, and guide. I wish my dad would be alive in good health to celebrate this together. Thanks to you and mami for your love and devotion to raising our beautiful family. Thanks to my dear Abuela Catalina and my beloved sister Dory for being always so supportive. Thanks to Manuel Fu for being like a brother to me while growing in Havana.

Thanks to my childhood and adulthood friends Franklin, Yuyo, and Willian. I still enjoy the dreams of our adventures in Cuba. An exceptional place in my gratitude is for Franklin and his mom Margarita, which under challenging times, gave me the warmth, love, and support in all possible ways a family can give to a son to thrive. I would never forget I have a family in Miami and a brother I look at as a light and example.

My gratitude also goes to my colleagues at the Cuban Neuroscience Center, where I was introduced to the neurosciences' fascinating world. To my friends not there anymore (Ale, Pollito, Itu, Charly, Pedrito, Yana, Roney, Nelson, Marlis, Eleina, Valia, Lester, Zoraida..... I'm talking about all of you guys), thank you.

I want to express my gratitude to my advisor Kennet Kreutz-Delgado. I could not recall in entire life as a student any other professor putting in his class the world of algebra in such a joyful harmony. Yes, I learned to see the world as an extension of the linear algebra lessons I took with Kreutz-Delgado, and that was indeed....amazingly illuminating. I've been lucky enough to have Ken as an advisor and benefit from his knowledgeable persona and humanity in every one of our interactions. To him, my deepest gratitude.

I'm also very grateful for having the excellent guidance and exceptional support of Arnaud Delorme, to whom I joined in 2014 for codeveloping the EEGLAB software at the Swartz Center for Computational Neurosciences.

A special acknowledgment goes to Dr. Scott Makeig at the Swartz Center for

Neurosciences. I'll always be grateful to Scott for opening the doors of the SCCN family to me. In Scott, I found a knowledgeable guide, a supportive supervisor, a colleague, and a friend. I can say with no regret that my Ph.D. would not have been possible without Scott's support and guidance. I hope he knows how he will always have a very special place in my thoughts and gratitude.

Thanks also to the wonderful SCCN family: Robert "Bob" Buffington, Ching-fu Chen, Dung Truong, Sidharth Sidharth, Tzy-Ping Jung, Masaki Nakanishi, Sheng-Hsiou "Shawn" Hsu, John Iversen, Mateusz Gola, Ying Wu, Luca Pion-Tonachini, Makoto Miyakoshi, and many others that came and went over these years as visitors scholars. To all of them, my sincere gratitude. Being at the SCCN for all these years was a deep professional but also personal transformative experience. At the SCCN, I met the extraordinary woman who is my wife today. What can I say of someone who makes me dream while awake? Words get lost trying to find the superlative but humble description of my love, gratitude, and admiration towards Johanna. I have so many reasons for being grateful to you, my dear....but here I would like to thank you for your unconditional and unwavering support during all these years while working towards the Ph.D. degree. My love and gratitude to you.

Chapter 2, in full, is a reprint of the material as it appear in Ramon Martinez-Cancino, Joseph Heng, Arnaud Delorme, Ken Kreutz-Delgado, Roberto C. Sotero, and Scott Makeig. "*Measuring transient phase-amplitude coupling using local mutual information.*" *NeuroImage* 185 (2019): 361-378. The dissertation author was the primary investigator and author of this paper. This work was supported by National Institutes of Health grant 5R01-NS047293-12 and by a gift from The Swartz Foundation (Old Field NY).

Chapter 3, in full, is a reprint of the material as it appear in Ramon Martinez-Cancino, Arnaud Delorme, Johanna Wagner, Ken Kreutz-Delgado, Roberto C. Sotero and Scott Makeig, 2020. *What can local transfer entropy tell us about phase-amplitude coupling*

in electrophysiological signals?. *Entropy*, 22(11), p.1262. The dissertation author was the primary investigator and author of this paper. This work was supported by National Institutes of Health grants 5R01-NS047293, R01MH123231 and by a gift from The Swartz Foundation (Old Field NY).

Chapter 4, in full, is a reprint of the material as it may appear in Ramon Martinez-Cancino, Arnaud Delorme, Ken Kreutz-Delgado, Scott Makeig. “*Computing phase amplitude coupling in EEGLAB: PACTools*” Proceedings of the 20th International IEEE Conference on BioInformatics and BioEngineering, October 26-28, 2020. The dissertation author was the primary investigator and author of this paper. This work was supported by National Institutes of Health grants R01-NS047293 (EEGLAB) and R01-EB023297 (Open EEGLAB Portal), and by a gift to UCSD from The Swartz Foundation (Old Field, NY).

Chapter 5, in full, is a reprint of the material as it appear in Ramon Martinez-Cancino, Arnaud Delorme, Dung Truong, Fiorenzo Artoni, Ken Kreutz-Delgado, Subhashini Sivagnanam, Kenneth Yoshimoto, Amitava Majumdar, Scott Makeig. " *The Open EEGLAB Portal Interface: High-Performance Computing with EEGLAB.*" *NeuroImage* (2020): 116778. The dissertation author was the primary investigator and author of this paper. This work was supported by the National Institutes of Health, U.S.A. (R01-NS047293, R24-MH120037, R01-MH123231) and the National Science Foundation (1458840), and by a gift from The Swartz Foundation (Old Field, NY).

VITA

- 2006 Bachelor in Nuclear Engineering, Higher Institute of Technologies and Applied Sciences, Havana, Cuba
- 2006–2010 Research Scientist, Cuban Neuroscience Center, Havana, Cuba
- 2009–2011 Lecturer on Physics, Technological University of Havana José Antonio Echeverría, Havana, Cuba
- 2009 Master of Science in Neurophysics and Neuroengineering, National Center for Scientific Research, Havana, Cuba
- 2010–2012 Research Scientist, Cuban National Bioinformatics Center, Havana, Cuba
- 2013–2014 Faculty Research Assistant/ Research Associate at Maryland Neuroimaging Center, College Park, Maryland, USA
- 2014–2020 Programmer Analyst II, Swartz Center for Computational Neuroscience, Institute for Neural Computation, University of California San Diego, San Diego, USA
- 2020 Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics, and Control), University of California San Diego, San Diego, USA

PUBLICATIONS

Peer-reviewed publications

Pernet C, **Martinez-Cancino R**, Truong D, Makeig S, Delorme A. *From BIDS-formatted EEG data to group results: a fully reproducible workflow with EEGLAB and LIMO EEG*. Submitted for publication in *Frontiers in Neuroscience*, section Brain Imaging Methods. October 2020.

Martinez-Cancino R, Delorme A, Wagner J, Kreutz-Delgado K, Sotero RC and Makeig S, 2020. *What can local transfer entropy tell us about phase-amplitude coupling in electrophysiological signals?*. *Entropy*, 22(11), p.1262.

Grove TB, Lasagna CA, **Martinez-Cancino R**, Pamidighantam P, Deldin PJ, Tso IF. *Neural oscillatory abnormalities during gaze processing in schizophrenia: Evidence of reduced theta phase consistency and inter-areal theta-gamma coupling*. *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*. 2020 Sep 1.

Martinez-Cancino R, Delorme A, Kreutz-Delgado K, Makeig S. *Computing phase amplitude coupling in EEGLAB: PACTools*. Proceedings of the 20th International IEEE Conference on BioInformatics And BioEngineering, October 26-28, 2020

Sivagnanam S, Yoshimoto K, Carnevale T, Nadeau D, Kandes M, Petersen T, Truong D, **Martinez-Cancino R**, Delorme A, Makeig S, Majumdar A. *Neuroscience Gateway enabling large scale modeling and data processing in neuroscience research*. In Practice and Experience in Advanced Research Computing 2020 Jul 26 (pp. 510-513).

Martinez-Cancino R, Delorme A, Truong D, Artoni F, Kreutz-Delgado K, Sivagnanam S, Yoshimoto K, Majumdar A, Makeig S. *The Open EEGLAB Portal Interface: High-Performance Computing with EEGLAB*. NeuroImage. 2020 Apr 11:116778.

Delorme A, Majumdar A, Sivagnanam S, **Martinez-Cancino R**, Yoshimoto K, Makeig S. *The Open EEGLAB portal*. In 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER) 2019 Mar 20 (pp. 1142-1145). IEEE.

Martinez-Cancino R, Heng J, Delorme A, Kreutz-Delgado K, Sotero RC, Makeig S. *Measuring transient phase-amplitude coupling using local mutual information*. NeuroImage. 2019 Jan 15;185:361-78.

Wagner J, **Martinez-Cancino R**, Delorme A, Makeig S, Solis-Escalante T, Neuper C, Mueller-Putz G. *High-density EEG mobile brain/body imaging data recorded during a challenging auditory gait pacing task*. Scientific Data. 2019 Oct 17;6(1):1-9.

Wagner J, **Martinez-Cancino R**, Makeig S. *Trial-by-trial source-resolved EEG responses to gait task challenges predict subsequent step adaptation*. Neuroimage. 2019 Oct 1;199:691-703.

Estrin S, **Martinez-Cancino R**, Makeig S, Gilja V. *Improving Classification Accuracy in Cortical Surface Recordings Using ICA-Based Features*. In 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2018 Oct 7 (pp. 3285-3288). IEEE.

Martinez-Cancino R, Diaz RC. *Dynamical Mean Field approximation of a canonical cortical model for studying inter-population synchrony*. Nature Precedings. 2011 Jan 24:1.

Sotero RC, **Martinez-Cancino R**. *Dynamical mean field model of a neural-glial mass*. Neural computation. 2010 Apr;22(4):969-97.

Sotero RC, Bortel A, **Martinez-Cancino R**, Neupane S, O'Connor P, Carbonell F, Shmuel A. *Anatomically-constrained effective connectivity among layers in a cortical column modeled and estimated from local field potentials*. Journal of Integrative Neuroscience.

2010 Dec;9(04):355-79.

Martinez-Cancino R, Sotero RC. *Modeling the effect of cytoplasm sol-gel transitions on magnetization changes during MRI diffusion experiments in brain gray matter*. International Journal of Bioelectromagnetism. 2008;10(4):269-80.

Non peer-reviewed

Wagner J, **Martinez-Cancino R**, Müller-Putz GR, Makeig, S (2018) *Error-related brain dynamics predict step adaptation in a challenging gait task*. Society for Neuroscience annual meeting, San Diego, USA

Wagner J, **Martinez-Cancino R**, Müller-Putz G, Makeig S. (2018) *Error-related brain dynamics predict step adaptation in a challenging gait task*. 3rd International Mobile Brain/Body Imaging Conference, Berlin, Germany

Wagner J, **Martinez-Cancino R**, Neuper C, Müller-Putz G, Makeig S. (2016) *Dynamic oscillatory coupling in the EEG is related to the gait cycle phase*. Salk Motor Systems Symposium, San Diego, USA

Ojeda A, **Martinez-Cancino R**, Kreutz-Delgado K, Valdes-Sosa P, Mishra J. *Local mutual information estimator of functional connectivity on EEG source sparse time series*. Presented at 25th Joint Symposium on Neural Computation, April, 2018, Pasadena, USA.

Martinez-Cancino R, Heng JA, Delorme A, Kreutz-Delgado K, Sotero RC, Makeig S. *Towards the estimation of phase-amplitude coupling temporal dynamics in neurophysiological signals: A modeling study*. Presented at the Society for Neurosciences meeting, Nov, 2017, USA

Pion-Tonachini L, **Martinez-Cancino R**, K. Kreutz-Delgado, S. Makeig. *Crowdsourced label learning for independent components of EEG data*. Presented at the Society for Neurosciences meeting, Nov, 2016. San Diego, Washington DC, USA

Martinez-Cancino R, Makeig S, Pernet C, Delorme A. *Hierarchical general-linear modeling of EEG data in EEGLAB and LIMO extension*. Presented at the Society for Neurosciences meeting, Nov, 2016. San Diego, USA

Wagner J, **Martinez-Cancino R**, Neuper C, Mueller-Putz G, Makeig S. *Dynamic phase-amplitude coupling in the EEG during gait adaptation*. Presented at the Society for Neurosciences meeting, Nov, 2016. San Diego, USA

Shackman A, Fox AS, Oler JA, Weinstein JS, **Martinez-Cancino R**, Oakes TR, Shelton SE, Smith JF, Pessoa L, Gamer M, Davidson RJ, Kalin NH (2015). *The extended amygdala is a key substrate for sustained anxiety*. Talk presented at the annual meeting of the Society of Biological Psychiatry (Symposium on Extended amygdala circuits in anxiety and addiction: cross-species molecular, anatomical, and functional insights), Toronto, Canada.

Riera JJ, **Martinez-Cancino R**, Sumiyoshi A, Nonaka H, Uchida T, Bosch J, Kawashima R, *Resting-state functional MRI for an animal model of Alzheimer disease*. Presented at the Society for Neurosciences meeting, Oct, 2012, New Orleans, USA

Martinez-Cancino R., Sotero R. Dynamical mean field model of a neural-glial mass. Presented at Neurosciences 2011, Nov, 2011, Washington DC, USA

Sotero R, **Martinez-Cancino R**, Shmuel A. *Modeling transcranial magnetic stimulation induced synaptic plasticity*. Presented at Neurosciences 2011, Nov, 2011, Washington DC, USA

Martinez-Cancino, R, Sotero R, *Astrocyte-induced synchronization as a relevant mechanism for epilepsy*. Presented at the congress Pharmacology Havana 2010, Dec 16, 2010, Havana, Cuba

Martinez-Cancino, R, Sotero R, *Dynamical mean field model of a neural-glial mass*. Presented at the Bernstein Conference on Computational Neurosciences, Oct 27 – Nov 1, 2010, Berlin, Germany

Martinez-Cancino R, Sotero R. *Modeling the effect of cytoplasm sol-gel transitions on magnetization changes during MRI diffusion experiments in brain gray matter*. Presented at the 3rd Scientific Conference Neurological Restoration, March 9-13, 2009, Havana, Cuba

Martinez-Cancino R, Sotero R. *Modeling the origin of the diffusion weighted MRI (DWMRI) signal from gray matter in multiple sclerosis*. Presented at the 1st International Workshop of NeuroImmunology, April, 2008, Varadero, Cuba, Cuban Journal of Pharmacology, vol. 42 (Sup 1): 22, 2008

Martinez-Cancino R, Sotero R, Sanchez-Bornot JM. *Analytical Model of Water Diffusion in Gray Matter*. Presented at the IV Cuban Congress and I Latin-American Workshop on Clinical Neurophysiology, March, 2008, Varadero, Cuba, Clinical Neurophysiology, 2008, doi: 10.1016/j.clinph.2008.04.168

ABSTRACT OF THE DISSERTATION

**Methods and tools for characterizing transient phase-amplitude coupling in
neural oscillations**

by

Ramon Martinez Cancino

Doctor of Philosophy in Electrical Engineering
(Intelligent Systems, Robotics, and Control)

University of California San Diego, 2020

Kenneth Kreutz-Delgado, Chair

Electroencephalography (EEG) is a fundamental technique for studying and understanding the brain through its oscillatory electrical outputs. Within the field of brain electrophysiology, phase-amplitude coupling (PAC) refers to the temporary interplay of activities in different frequency ranges, in which phase in a low-frequency band and amplitude in a high-frequency band are in some way dependent. The study of this phenomenon has been gaining attention due to its presence in normal and pathological brain processes in humans and across different mammalian species. In the quest to understand PAC's physiological role in the brain, the development of signal processing methods and software

that facilitate its study has become key to further analysis and advancement. Several methods have been proposed to estimate a measure of the PAC process, but few have addressed its temporal dynamics. Furthermore, no studies have addressed PAC dynamics while assuming its possible directional delay characteristics.

The goal of this dissertation is twofold: to provide new methods to estimate and characterize PAC dynamics, and to provide dedicated software with high-performance computing (HPC) support for estimating PAC from electrophysiological brain signals.

To begin with, I propose and validate a mutual information-based method for computing time-resolved PAC, MIPAC. Secondly, I characterize and outline the foundations for using the novel directed information theory measure, transfer entropy, to study delayed interactions from the phase to amplitude components in the PAC process. Following, I propose a new EEGLAB plug-in toolbox, *PACTools*, that allows for computing PAC in either continuous or event-related EEG data using any of five methods, including the newly developed MIPAC. *PACTools* operates seamlessly with EEGLAB, and features built-in access to free HPC resources at the Neurosciences Gateway (*nsgportal.org*). This feature is enabled by a second, newly developed EEGLAB plug-in, *nsgportal*, that interfaces EEGLAB with NSG directly from within EEGLAB on any personal computer. The tools and methods presented in this dissertation constitute a comprehensive framework aimed at facilitating the study of the PAC's role in the healthy and pathological brain.

Chapter 1

Introduction

Understanding the biological basis of the mind and consciousness is considered the last frontier in biological sciences [1]. The earliest account of humans referring to the brain date back to the Seventeenth Century B.C.: a papyrus describing the symptoms, diagnosis, and prognosis of patients with skull fractures (Fig. 1.1). Nevertheless, the desire to understand the source of our thoughts and behavior probably dates back to the very same origin of the homo sapiens. In the challenging and noble quest towards disentangling how the brain works, an interwoven challenging issue, in no way smaller, is demanding a scientific, political, and social undertaking. Let us begin a discussion on mental health.

In recent years there has been increasing acknowledgment on the gravity of the global mental health crisis, as evidenced by the inclusion of mental health in the United Nations Sustainable Development Goals [2; 3]. However, in the year 2020, the state of worldwide mental health is alarming: Depression remains a leading cause of disabilities, while suicide is the second leading cause of death among 15-29-year-olds. People with severe mental health conditions see their lifespan reduced as much as 20 years due to preventable conditions. On the economic side, the two most common mental health conditions, depression and anxiety, cost the global economy US\$ 1 trillion each year [2]. I have been deeply moved by the reality that comes with the diagnosis and prognosis of these ailments in developed and developing countries. My hope is to make a small

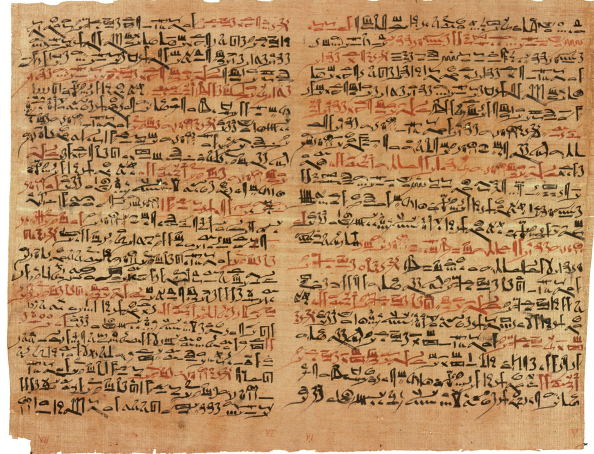


Figure 1.1. Plates vi & vii of the Edwin Smith Papyrus at the Rare Book Room. By Jeff Dahl. (<https://commons.wikimedia.org/w/index.php?curid=2854143>)

contribution towards understanding and solving these problems with this dissertation.

In this dissertation, my focus is on developing methods and tools to study and enable the analysis of a brain phenomenon -phase-amplitude coupling (PAC)- associated with a wide range of normal brain processes and pathologies, including Parkinson’s disease, epilepsy, and schizophrenia in humans. (e.g., [4–6]).

Before diving into the contributions of this dissertation in the effort of disentangling the role of PAC in the brain, it is convenient to outline the context in which this study’s contributions take place.

1.1 The human brain

The human brain is the largest, and without doubt, the most complex and fascinating organ of the human body. A masterpiece of biological and evolutionary engineering that allows our species to thrive, most of the time, towards progress. The brain is a key part of the central nervous system (CNS) and is encapsulated and protected by the head’s skull bones. At the cellular level, the brain is mainly composed of two large groups of cells: neurons and glia, with neurons outnumbering glia in a ratio of 10:1.

The CNS comprises four main types of glial cells: astrocytes, microglia, ependymal

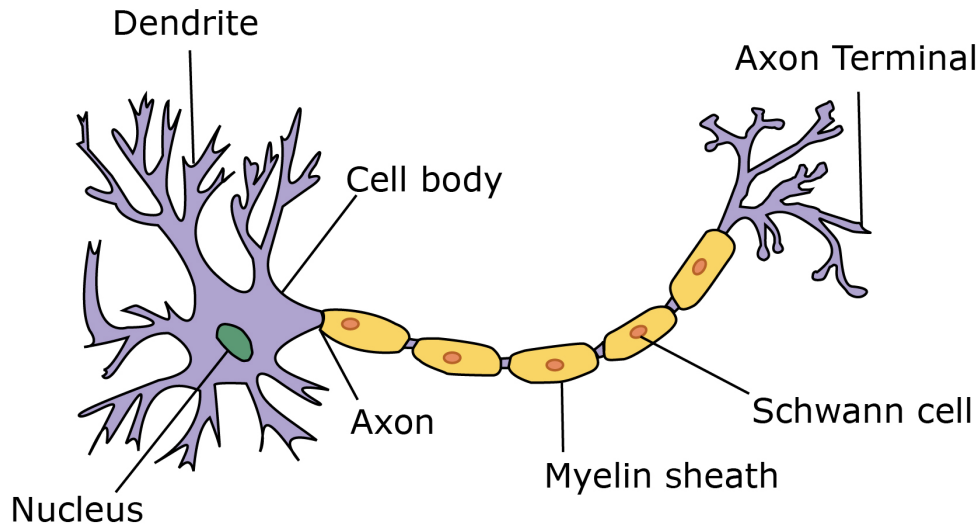


Figure 1.2. *Structure of a neuron.* Excitatory neurons share a common anatomical and physiological structure comprised mainly by four regions.(1) the cell body, (2) dendrites, (3) axon, and (4) presynaptic terminals. By user:Dhp1080 - "Anatomy and Physiology" by the US National Cancer Institute's Surveillance, Epidemiology and End Results (SEER) Program. CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=1474927>

cells, and oligodendrocytes. Astrocytes, the most abundant glial cells in the CNS, organize themselves in networks of interconnected cells and processes that regulate ion concentrations, and provide the neurons with essential nutrients. Astrocytes have been shown to shape and influence communication among neurons (I have pioneered the publication of biophysical models on this subject [7]). Other types of glial cells (e.g., Schwann cells) have been demonstrated to be responsible for aiding neural communication transmission.

The neuron is arguably the most critical cell of the brain. Individual neurons are either excitatory or inhibitory. Approximately 80% of the 100 billion neurons in the brain are excitatory, meaning they push or promote adjacent neurons to "*fire*". A typical neuron has four morphologically defined regions: (1) the cell body, (2) dendrites, (3) axon, and (4) presynaptic terminals (Fig. 1.2).

The neuron's cell body is a warehouse of genetic information, from where two types of cell processes emerge: several tree-like branch dendrites and a single, long cylindrical axon. Dendrites are responsible for collecting incoming information conveyed

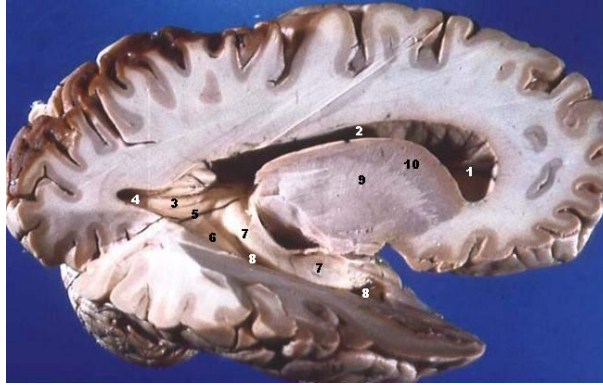


Figure 1.3. *Brain's gray and white matter.* Human brain right dissected lateral view showing the thin shelled gray matter layer covering the brain and the white matter underneath it. By John A Beal, PhD. Dept. of Cellular Biology; Anatomy, Louisiana State University Health Sciences Center Shreveport -<http://www.healcentral.org/healapp/showMetadata?metadataId=40566> (Internet Archive of file description page), CC BY 2.5, <https://commons.wikimedia.org/w/index.php?curid=886193>

from connecting neurons. In contrast, the axon conducts information from neuron to neuron through electrochemical pulses known as action potentials. Axons are wrapped by Schwann cells, which envelop sections of the axon with a regularly interrupted fatty layer of *myelin* to accelerate the transmission of the action potential. By the end of the axonal process, the axon is split into several fine branches that interface other neuron's specialized branches across inter-neural spaces know as *synapses*. Communication between the synaptic and postsynaptic neurons is based on the release and binding of biochemical messengers known as neurotransmitters, which can enact an excitatory or inhibitory influence on the neighboring neurons. While inhibitory cells have short processes restricted to small neuronal neighborhoods, excitatory neurons have remarkably long axons interconnecting neighboring and distant cells. This complex circuitry defines the anatomical division of the brain's gray and white matter (Fig. 1.3).

On a larger scale, the brain white matter is formed by axons interconnecting specialized brain regions, while the gray matter comprises the neuronal cell bodies that lie in the scaffolding provided by the astrocytic network. The synchronous activation and firing of clusters of neurons in the cortex gives rise to rhythmic electrical activity. In

addition, feedback loops between the surface-lining cortex and the Thalamus; a deeper brain structure that integrates sensory information and relays it to cortical regions [8], magnify and modulate this rhythmic activity [9]. Consequently, with proper techniques, electromagnetic fields associated with this phenomena can be detected from outside the skull and scalp. This phenomenon has served significantly in the study of normal brain functions and pathologies through the development of neuroelectrophysiology: the study of the brain through the analysis of its electrical activity. Advances in electrophysiological studies have brought attention to the signature patterns of the brain's electrical activity : Neural oscillations.

1.2 Looking into the brain through its electrical activity

The EEG is born

A prominent feature of electrical brain activity is the presence of rhythmic patterns, in both spontaneous and sensory-evoked cortical activity [9; 10]. The mammalian brain's spontaneous rhythms were first demonstrated in the 1870s by the Liverpoolian physiologist Richard Caton, and later investigated by Napoleon Cybulsky, Adolf Beck, and others [11]. However, the development of human electroencephalography (EEG) can be traced back to only one man: the German psychiatrist Hans Berger (1873-1941). The story and circumstances of the EEG discovery by Hans Berger were highly motivated by psychophysical research and by his personal experiences. During a year of military service in 1892, then 19-year-old, Berger was involved in a bizarre experience that inspired his life-long search for a connection between mind and brain. One morning while transporting artillery on horseback, Hans Berger fell to the ground just in front of the wheel of a heavy piece of artillery. Luckily, the horse-drawn battery stopped just before reaching the young Berger's head, saving him from certain death. The same evening, he received

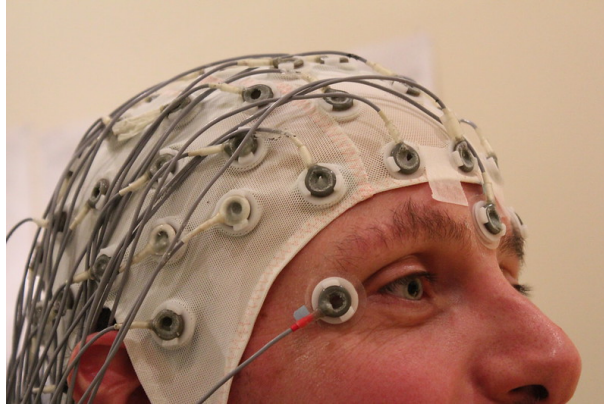


Figure 1.4. *EEG recording.* During an EEG recording session, electrodes consisting of flat metal discs are attached to the scalp. In some cases, a conductive gel is applied between the scalp and the electrode to improve the bioelectrical currents' conduction. The figure here shown a high-density EEG setup where the electrodes are closely spaced together. EEG electrodes are connected to an amplifier that amplifies the recorded potential and converts it into a digital signal suitable for analysis or visualization by a computer. By Chris Hope. CC BY 2.0 (https://www.flickr.com/photos/tim_uk/8135755109/)

a telegram from his father inquiring about his health. As he learned later on, that very morning his sister had the ominous feeling that something terrible had happened to Hans, and she rushed her father to contact him. Berger was convinced he and his sister had communicated by telepathy. This experience marked his life-long carrier in neurosciences [11].

In his pursuit to discovery, Hans Berger reported the first human EEG observation in 1924. This result was later documented in the article *Über das elektrenkephalogramm des menschen* in 1929 [12]. In modern day, EEG is still practiced under the same principles used by Berger, by recording the neural electrical fields through electrodes placed on the scalp (see Figure 1.4) -of course, the advancement and miniaturization of electronic elements have drastically improved EEG technology-, although invasive electrodes are sometimes used.

Rhythms of the brain

In the first documentation of this technique [12], Hans Berger reported the observation of what are known today as Alpha and Beta waves in EEG, which correspond to activity in the frequency bands corresponding to the ranges of 8-12 Hz and 12-30 Hz respectively. Since then, rhythmic activity in EEG has been widely reported and studied due to its ability to characterize normal and pathological brain states (for a review, see an excellent book on brain rhythms by Buzsaki [10]). It has been hypothesized that essential features of brain organization, such as activity-information retention, and local-global integration, are maintained by a hierarchical system of brain oscillations [10]; and that despite a 17,000-fold variability in brain volume across the mammalian species, irrespective of brain size, the management of multiple time-scales is supported by the same fundamental mechanisms [13]. An overwhelming amount of research indicates that the source of the rhythmic electromagnetic oscillations detected in EEG arises by the synchronous activity of patches of cortical pyramidal neurons in close interplay with thalamic structures [9; 10; 14].

Cross-frequency coupling and phase-amplitude coupling

It is now well accepted that brain oscillations often coexist and interact with each other, either within the same or across different brain structures, through the cross-frequency coupling (CFC) process. Theoretically, there are several ways in which the amplitude, frequency, and phase from two signals can interact. These different types of CFC have been described in the literature (e.g., [15]): (i) amplitude to amplitude, (ii) phase to phase, (iii) phase to frequency, and (iv) phase to amplitude. There is an increasing amount of evidence (see review in [15]) indicating that, among all these types of CFC, the fourth one, also known as phase-amplitude coupling (PAC) may be a widespread process of key importance for essential brain functions. Particularly, since slow oscillations are composed of synchronized neuronal states alternating between hyperpolarization (down-state) and depolarization (up-state), it has been proposed that CFC, and specifically, PAC, can support

neural syntactical rules. This mechanism facilitates more straightforward communication among brain areas when compared to the interpretation of long uninterrupted messages, and allows for the generation of rich combinations from a limited number of elements using minimal rules[16]. PAC has been observed among various frequency bands, in multiple brain regions, in different task conditions, and in multiple species (see Table 1 in [17] for review). PAC has also been related to a variety of psychiatric and neurological diseases(e.g., in epilepsy [18], Parkinson’s disease [19; 20], Alzheimer’s disease [21], schizophrenia [22; 23], obsessive-compulsive disorder [24], and mild cognitive impairment [25]), hence its indisputable importance.

Advances and current limitations of estimating phase-amplitude coupling

Several methods (e.g., [26–28]) have been proposed to estimate a magnitude describing the PAC process from electrical brain recordings. However, none has been established yet as the gold standard (the most important methods are reviewed in Chapter 2 and 3).

Most of the proposed methods to compute PAC follow a similar and straightforward pipeline. First, high and low-frequency band signals are extracted from the two frequency ranges in which the PAC coupling is to be assessed using band pass filters or time-frequency decomposition. Then, instantaneous phase and amplitude are extracted from the low and high frequency band respectively using the Hilbert Transform, or directly from the complex signal when time-frequency decomposition has been used. At this point, a measure assessing the magnitude of the coupling is computed from the two signals according to the method used. After the PAC measure is computed, a statistical analysis is conducted by comparing the estimated PAC value with a distribution of surrogate PAC values generated under a no-PAC assumption. Statistical significance follows by testing if the current PAC values fall within the distribution of no-PAC values.

Despite its relevance, PAC analysis and its physiological interpretation come with fundamental problems [29]. For example, spurious PAC may arise due to spectral correla-

tions promoted by common non-stationarities, and also the selection of the bandwidth towards computing the Hilbert transform may affect the PAC outcome [29]. These pitfalls and limitations have sparked a wave of different methods to approach PAC estimation. In addition to these critical confounds in the CFC analysis, I believe an important factor behind the diversity of recent methods proposed to compute PAC is the inability to efficiently manage significant limitations, like addressing temporal evolution and delayed interactions in the coupling process. These limitations are addressed in this dissertation.

Driven by the widespread study of the PAC phenomenon, most academic open-source software environments for analyzing electrophysiological signals (e.g., Fieldtrip [30], Brainstorm [31], MNE [32]) have implemented at least one PAC estimation method. A major contribution in this dissertation, is to provide a toolbox to compute PAC seamlessly integrated into the EEGLAB software through its plug-in capabilities. However, PAC computation may be highly computationally intensive when lengthy data is used; algorithm complexity may add to this problem. Thus, integrating PAC computing capabilities into major software should be hand in hand with developing means for handling big data and extensive computations. I address this limitation in this dissertation by creating software and infrastructure that facilitates access of PAC computing tools -and generally, to any functionality from EEGLAB- to high-performance computing resources. It is important to mention that EEGLAB is currently the most widely used software environment for analyzing EEG in the academic environment [33; 34], which ensures the contributions in this dissertation will have a unique amplifying platform to maximize its impact.

1.3 Information theory in neurosciences

The information theory field was born out of the genius and vision of its creator, Claude Shannon, to provide a framework to formalize the quantification, storage, and communication of information in communication systems. Following his landmark publica-

tion in this field, "*A Mathematical Theory of Communication*" [35], several manuscripts provided the foundations of much of the current use of information theory in neuroscience (e.g., [36–38]). Consequently, there is a long history of the application of information theory measures to neuroscientific systems. Recent developments in the field -specifically, directed information and estimators for pointwise quantities- have brought a renewed interest in its application to the neurosciences. One of the current dissertation goals is to address the estimation of directed temporal dynamics, including delayed interactions in the PAC estimation process. Information theory provides a suitable framework to approach this problem since statistical relationships between the frequency components (e.g., phase and amplitude) are naturally formulated within the probabilistic framework without assuming explicit models [39]. The specific development of directed measures in information theory, chiefly transfer entropy proposed by Schreiber [40], has also provided the possibility to approach Wiener's causality from a theoretical information framework, enabling subsequent expansions on, e.g., estimating delays constants and predictive information transfer [41]. Due to the advantages mentioned here, I chose to approach the estimation of directed dynamics of PAC using the information theory framework (see Chapter 2 and 3)). Due to the structure of the dissertation, comprising of freestanding chapters, introduction to relevant information theory measures is provided on each of the chapters where this framework is used (see Chapter 3 and 2).

1.4 Goal and specific aims

The main goal of this dissertation is to *develop analytical methods in the framework of information theory and computational tools to characterize PAC temporal dynamics in electrophysiological signals from the brain.*

The specific aims to ensure the fulfillment of the main goal are:

1. Developing and validating a method to estimate the temporal dynamics of PAC.

2. Developing and validating a method to characterize delayed interaction dynamics in PAC.
3. Designing, developing, implementing and validating a computational tool to calculate PAC from EEGLAB.

1.5 Chapters overview

Here I provide an overview of the chapters in this dissertation and the publications that resulted from each of them. Each chapter is freestanding, meaning that a reader interested in a specific chapter's contribution may go directly to it without reading the preceding material. However, chapters have been organized to emphasize the natural flow of their contributions to the dissertation goal.

Chapter 2: Measuring transient phase-amplitude coupling using local mutual information. This chapter proposes and validates a novel method for estimating time-resolved PAC from electrophysiological signals by using local mutual information: MIPAC. Both simulated and actual data are used to validate two methods to compute PAC: (1) Instantaneous MIPAC for continuous EEG data; and Event-related MIPAC for an assembly of data snippets corresponding to event-related EEG data.

The work in this chapter led to the following publication:

- **Martinez-Cancino R**, Heng J, Delorme A, Kreutz-Delgado K, Sotero RC, Makeig S. *Measuring transient phase-amplitude coupling using local mutual information*. NeuroImage. 2019 Jan 15;185:361-78.

Chapter 3: What can transfer entropy tell us about phase-amplitude coupling in electrophysiological signals? In this chapter, I characterize using a directed information theory measure, transfer entropy, to estimate PAC dynamics and estimate delays between the phase and amplitude components. Results from the application of transfer entropy to PAC estimation in both simulated and actual EEG data are discussed

along with the limitations which I consider necessary to address before recommending the use of this measure for PAC analysis.

The work in this chapter led to the following publication:

- **Martinez-Cancino R**, Delorme A, Wagner J, Kreutz-Delgado K, Sotero RC and Makeig S, 2020. *What can local transfer entropy tell us about phase-amplitude coupling in electrophysiological signals?*. Entropy, 22(11), p.1262.

Chapter 4: Computing phase amplitude coupling in EEGLAB: PACTools

Here I describe an EEGLAB plug-in, *PACTools*, that computes PAC in either continuous or event-related data using any of five methods for PAC estimation: Mean Vector Length Modulation Index [28], Kullback-Leibler Modulation Index [26], Phase-Locking Value [42], General Linear Model Modulation Index [27] and the Mutual Information PAC (MIPAC) introduced in Chapter 2. *PACTools* implement built-in access to online high-performance computing resources at the Neurosciences Gateway (*nsgportal.org*) through functionalities and software created under the development of the *nsgportal* EEGLAB plug-in. This development is also part of this dissertation and is introduced in Chapter 5.

The work in this chapter led to the following publication:

- **Martinez-Cancino R**, Delorme A, Kreutz-Delgado K, Makeig S. *Computing phase amplitude coupling in EEGLAB: PACTools*. Proceedings of the 20th International IEEE Conference on BioInformatics And BioEngineering, October 26-28, 2020

Chapter 5: The Open EEGLAB Portal Interface: High-Performance Computing with EEGLAB

Here I introduce an EEGLAB plug-in, *nsgportal*, that interfaces EEGLAB with online high-performance computing resources at the Neurosciences Gateway (*nsgportal.org*) directly from within EEGLAB running on MATLAB on any personal lab computer. Details on *nsgportal* implementation, use, and examples of implementing built-in HPC capabilities in EEGLAB plug-ins are presented and discussed. This plug-in

is used in the *PACTools* plug-in in Chapter 4 to implement access to high performance computing.

The work in this chapter led to the following publications:

- **Martinez-Cancino R**, Delorme A, Truong D, Artoni F, Kreutz-Delgado K, Sivagnanam S, Yoshimoto K, Majumdar A, Makeig S. *The Open EEGLAB Portal Interface: High-Performance Computing with EEGLAB*. NeuroImage. 2020 Apr 11:116778.
- Delorme A, Majumdar A, Sivagnanam S, **Martinez-Cancino R**, Yoshimoto K, Makeig S. *The Open EEGLAB portal*. In 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER) 2019 Mar 20 (pp. 1142-1145). IEEE.
- Sivagnanam S, Yoshimoto K, Carnevale T, Nadeau D, Kandes M, Petersen T, Truong D, **Martinez-Cancino R**, Delorme A, Makeig S, Majumdar A. *Neuroscience Gateway enabling large scale modeling and data processing in neuroscience research*. InPractice and Experience in Advanced Research Computing 2020 Jul 26 (pp. 510-513).

Chapter 2

Measuring Transient Phase-Amplitude Coupling Using Local Mutual Information

2.1 Abstract

Here we demonstrate the suitability of a local mutual information measure for estimating the temporal dynamics of cross-frequency coupling (CFC) in brain electrophysiological signals. In CFC, concurrent activity streams in different frequency ranges interact and transiently couple. A particular form of CFC, phase-amplitude coupling (PAC), has raised interest given the growing amount of evidence of its possible role in healthy and pathological brain information processing. Although several methods have been proposed for PAC estimation, only a few have addressed the estimation of the temporal evolution of PAC, and these typically require a large number of experimental trials to return a reliable estimate. Here we explore the use of mutual information to estimate a PAC measure (MIPAC) in both continuous and event-related multi-trial data. To validate these two applications of the proposed method, we first apply it to a set of simulated phase-amplitude modulated signals and show that MIPAC can successfully recover the temporal dynamics of the simulated coupling in either continuous or multi-trial data. Finally, to explore the use of MIPAC to analyze data from human event-related paradigms, we apply it to an

actual event-related human electrocorticographic (ECoG) data set that exhibits strong PAC, demonstrating that the MIPAC estimator can be used to successfully characterize amplitude-modulation dynamics in electrophysiological data.

2.2 Introduction

Since the first observations of human brain (8-12 Hz) alpha waves by Hans Berger in 1929 [12], additional modes of rhythmic or semi-rhythmic cortical field activity have since been observed, including characteristic activities in the (1-4 Hz) delta, (4-8 Hz) theta, (12-30 Hz) beta and (above 30 Hz) gamma frequency bands. For many decades, the study of the spectral dynamics of cortical field oscillations was limited to analysis of phenomena occurring within single frequency bands. Over the last few years, the growing evidence that oscillations at different frequencies interact with each other has gathered attention, leading to heightened interest in cross-frequency coupling (CFC) in brain dynamics. Since high-frequency field activity is often more topologically localized than low frequency rhythms [9], CFC could be a mechanism for integrating information across spatial as well as temporal scales.

Although CFC could refer to any possible interaction between frequencies, phases and amplitudes of oscillatory phenomena [43], most experimental work has focused on three types of CFC: amplitude-amplitude coupling (AAC) or comodulation, phase-phase coupling (PPC) including bicoherence, and phase-amplitude coupling (PAC). Among them, PAC has attracted increasing interest given the growing amount of evidence of its potential role in brain information processing and its changes under pathological conditions including epilepsy [19; 20]. In PAC, the instantaneous amplitude of a higher frequency band within a signal is modulated by (or otherwise linked to) the instantaneous phase of a lower-frequency band of the same (or a different) signal.

PAC has been observed between various frequency bands, in multiple brain regions,

in different task conditions, and in multiple species. Table 1 illustrates the diversity of recent reports. In its diversity, PAC has shown that it can reveal unsuspected relationships, e.g., the intriguing PAC found between the basal gastric rhythm and alpha band power in cortex [44].

If PAC is a functional aspect of normal brain dynamics, then abnormal PAC could be a cause or symptom of unhealthy brain function. Associations between brain disease and PAC have been found in epilepsy [18], Parkinson’s disease [19; 20], Alzheimer’s disease [21], schizophrenia [22], obsessive-compulsive disorder (OCD) [24], and mild cognitive impairment (MCI), [25]. This makes PAC estimation of interest for clinical studies.

Several methods have been proposed to measure PAC. However, none is currently a gold standard (see Table 2.1). The three measures most often cited in the PAC literature are: the Mean Vector Length Modulation Index (MVLmi) [28], the Kullback-Leibler Modulation Index (KLmi) [26], and the General Linear Model Modulation Index (GLMmi) [27].

The data processing pipeline for these methods share a common beginning. A first step is to band-pass filter the signal or signals of interest to isolate the frequency ranges to be investigated. Then, an analytical signal is obtained for each frequency band using the Hilbert transform, from which instantaneous phase (of the low-frequency modulatory signal) and instantaneous amplitude (of the high-frequency modulated signal) are obtained. These time series are used then to compute a PAC measure as explained in detail below. To compute statistical significance, the collected sets of phases and/or amplitudes are shuffled many times to create surrogate data sets. Significance is then estimated by determining whether the PAC measure for the actual data belongs or not to the distribution of PAC measures computed on the surrogate data [26; 51]. The same process can be applied when PAC is assessed between low- and high-frequency dynamics of two different signals, e.g. between different electrophysiological recording channels or separated source activities in different brain areas.

Table 2.1. *Example of studies on Phase-Amplitude Coupling.* DBS: Deep Brain Stimulation. LFP: Local field potential. MEG: Magnetoencephalography. MVLmi: Mean Vector Length Modulation Index. KLmi: Kullback-Leibler Modulation Index. PLV: Phase Locking Value. DPAC: Direct PAC estimator. SI: Synchronization Index. PMI-PAC: Permutation Mutual Information based Phase-Amplitude Coupling

Study	Freq. Band	Brain Area	Modality	Study Focus	Species	PAC Method
Canolty et al. 2006 [28]	θ - high- γ	Neocortex	ECoG	Perception Memory Motor	Human	MVLmi
Cohen et al. 2009 [45]	α - low- γ	Nucleus Accumbens	DBS electrodes	Reward	Human	MVLmi
Tort 2009 et al. [4]	θ - γ	Hippocampus	LFP	Learning	Rat	KLmi
Voytek 2010 et al. [46]	θ/α - high- γ	Posterior cortex	ECoG	Perception	Human	PLV
Yanagisawa 2012 et al. [47]	α - high- γ	Sensorimotor cortex	ECoG	Motor	Human	SI
Spaak et al. (2012) [5]	α - low- γ / high- γ	Visual cortex	Laminar electrodes	Spontaneous activity	Monkey	KLmi
Szczepanski et al. 2014 [48]	δ/θ - high- γ	Fronto-parietal cortex	ECoG	Attention	Human	PLV
Florin et al. (2015) [49]	$\delta/\theta/\alpha$ - low- γ / high- γ	Resting state networks	MEG	Spontaneous activity	Human	DPAC
Sotero et al. (2015) [50]	$\delta/\theta/\alpha - \beta$ / low- γ / high- γ	S1	Laminar electrodes	Spontaneous activity	Rat	MVLmi
Cheng et al. (2017) [6]	θ - low- γ	Hippocampus CA3, CA1	LFP	Spontaneous activity	Rat	PMI-PAC

A common drawback of most of the existing PAC measures is their need for a large number of data points to compute a robust PAC estimate. A typical approach is to perform the PAC calculation using a semi-arbitrary time window [26; 27; 52]. The minimum length of this window is constrained by the frequency of the lower (phase) frequency of interest; at that frequency the analysis window should be at least one full cycle in length. However, a recent study [26] has proposed that because of the sensitivity to noise of existing PAC measures, the number of cycles needed to compute a reliable PAC estimate may be 200 cycles or more.

PAC was first measured across blocks of continuous data, providing relatively low sensitivity to changes in PAC across time. Subsequent work [53] tried to fill this gap by applying both automated and supervised heuristics to determine the frequencies involved in the coupling and by a combination of existing PAC measures to a moving time window. This method has been proven useful for characterizing dynamics of electrophysiological processes supporting cognition. However, this method fails to detect CFC in different scenarios; for example, when the amplitude time series does not have prominent components at the frequencies at which it is coupled [53].

Voytek *et al.* [52] addressed the lack of temporal resolution by computing PAC for latencies within multiple trials time locked to a set of similar events, thereby introducing Event-Related Phase Amplitude Coupling (ERPAC). In this approach, a variant of the GLMmi PAC estimator was applied in an event-related manner. However, this approach still cannot provide a trial-by-trial description of PAC variability and requires many experimental events that may be presumed to produce equivalent dynamics.

To disentangle the role(s) of PAC in neurophysiology, the question of *when* it occurs is as central as *where* or *how*. As suggested by Lizier [39], local or pointwise information theoretic measures can help answer this question. In this work we propose a new approach to PAC estimation, MIPAC, based on a local mutual information (MI) measure, that allows a dynamic description of PAC within a relatively short time window. The new method is

first tested on simulated and then on actual human event-related electrocorticographic (ECoG) data. We also compare MIPAC estimates with the results of other PAC estimators.

The manuscript is organized as follows: In Section 2.3, we describe the data on which we test the new method, as well as the required data processing. Then, in Section 2.3.1 we comment on the estimation of the instantaneous phase and amplitude in the signal of interest. Next, we give an introduction to traditional PAC measures and briefly review three current PAC estimation methods: Mean Vector Length Modulation Index (Section 2.3.2), Kullback-Leibler Modulation Index (Section 2.3.2) and Modulation Index based on a General Linear Model (Section 2.3.2). Following this, we present an introduction to Information-Theoretical measures and then introduce the MIPAC approach. Section 2.4 presents validation results of the new methods on simulated data and its application to human electrophysiological data. In Section 2.5 we discuss the results, some pitfalls and caveats of the proposed method, and its possible uses and extensions.

2.3 Methods

2.3.1 Ethics statement

Ethics statement: The patient data we study here were recorded during monitoring of implanted ECoG electrode grid and strip signals conducted during planning for surgery for epilepsy. The patient participated in a purely voluntary manner, after providing informed written consent, under experimental protocols approved by the Institutional Review Board (IRB) of the University of Washington (# 12193). All patient data were anonymized according to IRB protocol, in accordance with HIPAA mandates. These data originally appeared in the manuscript, “Spontaneous Decoding of the Timing and Content of Human Object Perception from Cortical Surface Recordings Reveals Complementary Information in the Event-Related Potential and Broadband Spectral Change”, published in *PLoS Computational Biology* in 2016 [54].

Experimental setting

A single data set (from subject *mv* in the data reported in [54]) was selected to demonstrate the methods proposed in this manuscript. The patient was clinically implanted with platinum subdural electrode grids and strips (Ad-Tech, Racine, WI) in frontal, parietal, temporal, and occipital cortical areas for extended clinical monitoring and localization of seizure foci. Lateral frontoparietal ECoG grid signals were not used in the analysis; we focused on signals from one strip located over interior temporal cortex. The electrodes were 4 mm in diameter (2.3 mm exposed), with 1 cm inter-electrode spacing, and were embedded in silastic. Electrode locations relative to gyral surface anatomy were determined by projecting the post-implant CT to the pre-operative axial T1 magnetic resonance head image using the normalized mutual information approach in SPM, and the CTMR package of Hermes *et al.* [55]. Cortical surface mesh reconstruction was performed using Freesurfer [56].

During the experiment, the subject was presented with a sequence of images of either a face ('face stimulus') or house ('house stimulus') in random sequence. There were three runs of 100 presentations each (50 houses and 50 faces in random order). Each stimulus presentation lasted 400 ms and was followed by a blank inter-stimulus interval of 400 ms. The ECoG potentials were measured with respect to a scalp reference and ground, subjected to recording system-imposed band-pass filtering between 0.15 Hz and 200 Hz, and sampled at 1000 Hz per channel. The protocol is described in more detail in [54].

Data preprocessing

Data processing, PAC computation and analysis were performed using EEGLAB functions [57] and custom scripts written in MATLAB (The Mathworks, Inc.). First, segments in the data containing strong non-brain artifact (e.g. high-frequency muscle noise and other irregular artifacts) as identified by visual inspection were removed. A common-average reference computation involving all channels, was applied to the

remaining data. The data were then resampled to 512 Hz and 60-Hz line noise was suppressed using a Hamming-windowed (sinc) FIR notch filter [order 847] (EEGLAB function *pop_eegfiltnew.m*) to minimize activity between 58 Hz and 62 Hz and between 118 Hz and 122 Hz (i.e., at the fundamental and first harmonic of the 60 Hz line noise). MATLAB scripts and functions for computing and visualizing different PAC measures in the EEGLAB environment have been collected into a toolbox developed by the authors (the Event-Related Phase-Amplitude Coupling Toolbox, ERPACT) that is now available from the EEGLAB Extension manager ¹ and from a GIT repository ².

Extracting phase and amplitude time series

As we mentioned earlier, most PAC estimation methods begin with computing phase and amplitude time series from the data. When analyzing PAC within a single signal, the original raw signal S_{raw_t} is first band-pass filtered to isolate the two frequency ranges in which the PAC coupling is to be assessed, respectively centered on a lower central frequency f_{phase} for the phase time series, and a higher central frequency f_{amp} for the amplitude time series. In the case of analyzing PAC between two signals the process will be similar to the one described, but extracting one frequency range from each of the signals under analysis.

Due to the uncertainty as to the ideal bandwidth for applying the Hilbert transform, the criteria for selecting the bandwidth for band-pass filtering diverges among authors. Usually, the bandwidth used in constructing the phase time series is required to be sufficiently narrow to derive a meaningful phase measure [29]. This requirement is often met. Meanwhile, when selecting the bandwidth used to obtain the amplitude time series, side-bands resulting from the amplitude-modulation (AM) process should be included within the measured bandwidth. Disregarding this criteria may result in inaccurate PAC estimates [29].

¹https://scn.ucsd.edu/wiki/Plugin_list_process

²https://bitbucket.org/ramonmc/pop_pac

In the current work, the bandwidth of the band-pass filter used to estimate the phase time series was $f_{PhaseBand} = [f_{phase} - 1Hz, f_{phase} + 1Hz]$, while for the amplitude time series the bandwidth was $f_{AmpBand} = [f_{amp} - (f_{phase} + 1Hz), f_{amp} + (f_{phase} + 1Hz)]$ to include the AM side-bands. Below, we denote by S_φ and S_A the signals obtained by band-pass filtering the signal(s) of interest in the ranges $f_{PhaseBand}$ and $f_{AmpBand}$ respectively.

To filter the data within these frequency bands, a combination of high-pass and low-pass FIR filters were coded using EEGLAB function *pop_eegfiltnew.m*. Analytic signals were then obtained by applying the Hilbert transform to the S_φ and S_A signals. Next, instantaneous phase φ_t and amplitude A_t time series were estimated from the analytic signals of S_φ and S_A respectively.

2.3.2 Traditional PAC measures

In this section we briefly review some of the methods most commonly used to estimate PAC. Later in the manuscript, we will compare results of these methods to results of the proposed MIPAC measure.

Mean Vector Length Modulation Index (MVLmi)

The method proposed by Canolty *et al.* [28] relies on generation of the composite values z_t in the complex plane, constructed by combining the instantaneous phase of the lower frequency signal (φ_t) and instantaneous amplitude of the higher frequency signal (A_t) as in Eq. 2.1.

$$z_t = A_t \cdot e^{i\varphi_t} \quad (2.1)$$

The distribution of the composite vectors z_t in the complex plane is then analyzed. If there is no PAC, since there is no affinity between a specific value of phase and amplitude, the distribution should be nearly radially symmetric. However, if the local

mean magnitude A_t is higher at some specific phase (or phases), then PAC is present and the phasor representation of the phase-amplitude distribution will exhibit a 'bump' at these phase values. The length of the composite mean vector obtained in Eq. 2.2 is used as a measure of phase-amplitude coupling. Here, N_c denotes the number of composite values averaged in the interval $t = [1, T]$. Alternatively, some authors [27] use in their PAC analysis the z -score of the MVLmi in Eq. 2.2 obtained from a surrogate data analysis (see Section 2.3.7)

$$MVLmi = \left| \frac{1}{N_c} \sum_{t=1}^T z_t \right| \quad (2.2)$$

Kullback-Leibler Modulation Index (KLmi)

Tort *et al.* [26] pointed out that the MVLmi approach assumes that coupling only occurs at one preferred phase and thus may perform poorly for more complex coupling relationships. To remedy this, they proposed a different approach. First, the instantaneous phase time series φ_t is subdivided into an arbitrary fixed number of bins N_{bin} and the mean of the instantaneous amplitude A_t is computed for each j -bin $j = \{1 \dots N_{bin}\}$, here denoted by $\langle A_t \rangle_{\varphi}(j)$. These mean values are then divided by the sum of all the amplitudes over the bins, giving a fractional value $P(j)$ for each bin j (Eq. 2.3).

$$P(j) = \frac{\langle A_t \rangle_{\varphi}(j)}{\sum_{k=1}^{N_{bin}} \langle A_t \rangle_{\varphi}(k)} \quad (2.3)$$

As $P(j) \geq 0 \forall j$ and $\sum_{j=1}^{N_{bin}} P(j) = 1$, P meets the criteria for a probability distribution function. If there is no coupling, this distribution should be uniform. If there is coupling, the deviation of this distribution from uniformity will be proportional to the strength of the coupling. This deviation is quantified by computing the Kullback-Leibler divergence (D_{KL}) between P and a uniform distribution U . The divergence measure is used to define

the KLmi (Eq. 2.4), where the $\log(N_{bin})$ term is used to reduce the dependency of the measure on the number of bins.

$$KLmi = \frac{D_{KL}(P \parallel U)}{\log N_{bin}} \quad (2.4)$$

$$with \ D_{KL}(P \parallel U) = \sum_{k=1}^{N_{bin}} \log \frac{P(k)}{U(k)} \quad (2.5)$$

Modulation Index based on General Linear Model (GLMmi)

The use of correlation to compute PAC was first proposed by Bruns and Eckhorn [58]. In this method, also referred as envelope-to-signal correlation (ESC), correlation between the time series (band-pass filtered in the range of the frequency of the phase of interest) and amplitude in the range of the higher frequency of interest was used as a measure of PAC. Later, Penny *et al.* [27] pointed out Bruns and Eckhorn ESC inability to capture the PAC phenomena in all instances (e.g. ESC can be confounded with amplitude co-modulation). They took, nonetheless, this measure as a starting point of their own approach. They first proposed to estimate PAC by means of the correlation between the amplitudes and the *cosines* of the phase signals, but pointed to the inability of this approach to detect coupling at the so called ‘null phases’ (e.g. $\cos(1/4)2\pi = 0$). To overcome this limitation, they proposed an approach using the general linear model (GLM) framework as in Eqs. 2.6 and 2.7. Here, the PAC estimate is inferred from the portion of variance explained by the linear model.

$$A_t = X\beta + e \quad (2.6)$$

$$\text{with } X = \begin{vmatrix} \cos \varphi_{t_1} & \sin \varphi_{t_1} & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cos \varphi_{t_{max}} & \sin \varphi_{t_{max}} & 1 \end{vmatrix} \quad (2.7)$$

In Eq. 2.6, β is the regression coefficient matrix and e is additive Gaussian noise. The design matrix X (Eq. 2.7) is composed of three columns; the first two columns are the *sine* and *cosine* of the time series φ_t respectively, and the third is a constant vector of ones. This design effectively circumvents the limitations of the previous correlation-based method.

Event-Related Phase Amplitude Coupling (ERPAC)

The GLMmi method proposed by Penny *et al.* (see Section 2.3.2) was modified by Voytek *et al.* [52] to allow estimation of PAC dynamics across a set of brief event-locked data epochs in event-related experiments.

To apply this method, the time series φ_t and A_t are broken into time windows of equal length, time-locked to the onset of equivalent experimental events. Times series for both magnitudes form a matrix with dimensions equal to the number of trials by the number of latencies. The method then follows as to the GLMmi method 2.3.2 by computing the PAC modulation index for each successive latency across trials. This way one can obtain a latency-resolved description of the evolution of PAC following a set of similar events.

2.3.3 Information-theoretic measures

Next, we will present an overview of the information-theoretic measures relevant to the approach proposed in this work. Let us start by considering two discrete random variables X and Y with sets of values x and y respectively, probability distributions $p(x)$,

$p(y)$, conditional probabilities $p(x | y)$, and joint distribution $p(x, y)$. A central quantity in Information Theory is the Shannon Entropy (H) (Eq. 3.1). The quantity H represents the average amount of uncertainty associated with a measurement x of the random variable X :

$$H(X) = - \sum_x p(x) \log_2 p(x) \quad (2.8)$$

Shannon Entropy can also be defined as the average Shannon information content $h(x)$ (Eq. 2.9) over each outcome x of the random variable X used to build the probability distribution function (pdf) $p(x)$ (Eq. 2.10):

$$h(x) = - \log_2 p(x) \quad (2.9)$$

$$H(X) = E_X[h(x)] \quad (2.10)$$

By convention, base 2 or e logarithms are used, giving units of bits or nats respectively.

By generalizing Shannon's definition of entropy to two random variables X and Y , the joint entropy can be defined as in Eq. 3.2, and represents the expected uncertainty of their joint distribution [39]. Alternatively, Eq. 3.2 can be written as a function of the Shannon information content as in Eq. 2.12.

$$H(X, Y) = - \sum_{x,y} p(x, y) \log_2 p(x, y) \quad (2.11)$$

$$H(X, Y) = E_{X,Y}[h(x, y)] \quad (2.12)$$

The conditional entropy of X given Y is defined as the average uncertainty about

x that remains when the value of y is known:

$$H(X | Y) = - \sum_{x,y} p(x, y) \log_2 p(x | y) \quad (2.13)$$

Using the definition of entropy in Eq. 3.1, we can formalize the mutual information (MI) between the random variables X and Y as a non-negative and symmetric measure defined in Eqs. 3.4 and 3.5:

$$I(X, Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \geq 0 \quad (2.14)$$

$$= H(X) - H(X | Y) \quad (2.15)$$

This quantity can be interpreted either as the average reduction in uncertainty about X given knowledge of the value of Y , or as its name suggests, as the amount of information shared by the two variables. Note that Eq. 3.4 is equal to zero if and only if X and Y are independent, which means $p(x, y) = p(x)p(y)$. Therefore, the mutual information is a measure of dependency (both linear and nonlinear) between the two random variables X and Y [59].

2.3.4 Defining local mutual information

In Eq. 2.12 we saw that the joint entropy of the random variables X and Y can be seen as the average of the joint Shannon information content $h(x, y)$ over its realizations x and y respectively. Given this interpretation, several authors have used the terms pointwise (particularly in natural language processing) or *local* information theoretic measures to describe quantities such as those defined in Eqs. 2.9 or 3.9 [39; 60–63]. These *local* information theoretic measures characterize quantities from a specific subset of measurements x and y of the random variables X and Y , rather than the associated average

measure computed over all available data [39]. An interesting discussion on how frequently used information-theoretic measures can be viewed as averages over their associated local measures can be found in [39]. As an example of this, local MI values $i(x, y)$ (Eq. 3.9) may be averaged to compute overall MI $I(X, Y)$ (Eq. 3.10):

$$i(x, y) = \log_2 \frac{p(x | y)}{p(x)} \quad (2.16)$$

$$I(X, Y) = E_{X, Y} [i(x, y)] \quad (2.17)$$

Negative local MI values might result if $p(x | y) < p(x)$ in Eq. 3.9. Thus, in contrast to the averaged quantity, which cannot take negative values, local MI values can be either negative or positive. These negative values may be interpreted as misinformative since the expectation of observing x is lowered by knowing y [39].

In Eq. 3.9, the local value of $i(x, y)$ is obtained by evaluating the pdf $p(x)$ and $p(x | y)$ at a set of local observations (x, y) . Meanwhile, the pdf here are defined over all the observations, both local and non-local to the observation under analysis. The term *local*, though, can be misleading when dealing with time series since it does not imply locality in the time-sense. Notice that until now, while introducing the basic information-theoretic measures and local MI, we have been dealing only with discrete random variables. This is, without assuming any type of temporal structure within the samples in question. This makes the *local* sense to be solely based upon the sample properties, e.g. distance between sample magnitudes. If a time dependency is assumed, as in the case of a time series, we will not be dealing anymore with a statistical sample, but the definition of *local* will remain as in the case of discrete random variables. Thus, the term *local* is not in relationship to the time structure of the process. Because of this, in the case of real time series, points locally close to the sample in the pdf are not necessarily local to the observation in the time sense. Thus, the stationarity of the time series used to build the pdf should be considered

carefully. In fact, the validity of the application to time series of the information theory measures introduced until now is under the assumption of stationarity or quasistationarity over time. We will return to this point in the next sections.

2.3.5 Estimating local mutual information

There is a long history of application of information theory methods in neurosciences, specially MI (for a review see [39; 64]). Although several methods for estimating MI have been proposed, the so called binning method remains the most commonly used approach. In this method, the supports of X and Y are partitioned into an arbitrary, limited number of bins, for which the joint probabilities are estimated by counting the number of samples in each bin. However, this method inevitably leads to serious bias problems [65–68]. This drawback can be alleviated to some extent by recent advances in development of MI nearest-neighbour estimators such as the one proposed by Kraskov, Stogbauer, and Grassberger (KSG) [59], together with a rigorous statistical test using appropriate surrogate data. Although computationally expensive for large amounts of data, the KSG estimator constitutes an effective non-parametric estimator of MI that is data efficient (resolve structures down to the smallest possible scales), adaptive resolution (binning scale changes according to the underlying data point density), and has minimal bias [59] (in consequence, it may have large variance).

Before presenting the local mutual information estimator, let us first introduce the KSG estimator and then move on to the local estimator. The KSG estimator was obtained by modifying and extending the nearest-neighbors-based Kozacheko-Leonenko estimator of entropy [69]. Instead of estimating directly the pdf needed to compute the MI, the KSG estimator makes use of the distances of each sample to its k^{th} nearest neighbor to count the neighbors in the marginal spaces, then plugs in these values in the MI estimator. To elaborate on this, let's assume a metric given in the space spanned by the discrete random variables X , Y and the joint space $Z = (X, Y)$. Then, the neighbors of each

point $z_i = (x_i, y_i)$ are ranked by the distance $d_{i,j} = \|z_i - z_j\| : d_{i,j_1} \leq d_{i,j_2} \leq \dots$. Similar rankings are done in the subspaces X and Y . While any norm can be used for these subspaces, the maximum norm is used for the joint space (Eq.2.18) [59]. At this point, marginal variables, or alternatively the norms, can be scaled to make them comparable. Additionally, although not strictly required, if the distributions are very skewed and/or rough, it might be a good idea to transform them such as to become more uniform in order to reduce errors in general [59].

$$\|z - z'\|_z = \max \{ \|x - x'\|, \|y - y'\| \} \quad (2.18)$$

Next, the k^{th} nearest neighbor for each point z_i , a distance $\varepsilon(i)/2$ away, is computed. Let's denote by $\varepsilon_x(i)/2$ and $\varepsilon_y(i)/2$ the distances between the same points projected into the subspaces X and Y respectively. As the maximum norm is used in Eq. 2.18, it becomes obvious that $\varepsilon(i) = \max \{ \varepsilon_x, \varepsilon_y \}$. In the KSG estimator used in our work, the number of points $n_x(i)$ and $n_y(i)$ are estimated by counting the points within the strips defined by $(x_i - \varepsilon(i)/2, x_i + \varepsilon(i)/2)$ and $(y_i - \varepsilon(i)/2, y_i + \varepsilon(i)/2)$ respectively. The process of computing the number of neighbors $n_x(i)$ and $n_y(i)$ of point z_i in the neighborhood defined by its k^{th} neighbor is depicted in Fig. 5.1.

After computing the number of neighbors n_x and n_y for each z_i , the estimate for MI is given by the first KSG estimator (Eq. 3.13).

$$I(X, Y) = \psi(k) - \langle \psi(n_x + 1) + \psi(n_y + 1) \rangle + \psi(N) \quad (2.19)$$

In Eq. 3.13, N is the number of samples and ψ denotes the digamma function ($\psi(x) = \Gamma(x)^{-1} \frac{d\Gamma(x)}{dx}$). MI values $I(X, Y)$ are returned in nats. By using this approach, pdf evaluations are bypassed and the average measure $I(X, Y)$ can be estimated for each observation from the nearest neighbors (counts n_x and n_y) in the marginal spaces. Taking the KSG estimator (Eq. 3.13) as a starting point, Lizier [39] proposed to obtain the local

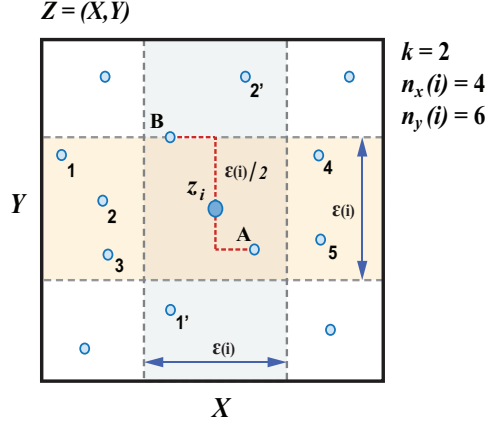


Figure 2.1. Schematic of Z -space showing the $k = 2$ nearest-neighbors search for data point z_i . Subspaces X and Y are represented in the horizontal and vertical axes respectively. The two nearest samples to z_i are **A** and **B**. Red dotted lines show the distance in the marginal spaces from the samples **A** and **B** to the sample z_i . Sample **B** is the $k = 2$ neighbor and its maximum marginal distance to the sample z_i ($\varepsilon/2$) defines the width of the marginal neighborhoods for X and Y , represented by the vertical and horizontal blue and orange stripes. Four sample points ($1'$, $2'$, **A**, **B**) are within distance $\varepsilon/2$ in the marginal space of X ($n_x(i) = 4$); six sample points (1 , 2 , 3 , 4 , 5 , **A**) are within distance $\varepsilon/2$ in the marginal space of Y ($n_y(i) = 6$). Several other samples (unlabeled) are outside the $k = 2$ neighborhood.

quantities by unrolling the expectation values ($\langle \dots \rangle$) and computing the nearest neighbor count only at the given (x, y) (Eq. 3.14):

$$i(x, y) = \psi(k) - \psi(n_x + 1) - \psi(n_y + 1) + \psi(N) \quad (2.20)$$

This estimator of local MI has been previously treated as a 'time-resolved estimator' in [70]. If implemented in its naive version using nested loops through all N points, the KSG algorithm spends most of the CPU time searching for neighbors. This lead to a complexity $O(N^2)$ [59]. Although this may be acceptable for small data sets (say $N \leq 3500$), strategies including fast nearest neighbor search algorithms are needed to deal with higher dimensionality. Here we used the naive version.

2.3.6 Estimating PAC using local mutual information

Here, we apply the local MI estimation approach presented in Section 2.3.5 to calculate PAC in brain electrophysiological signals. As mentioned before, to calculate local MI, as well as other information-theoretical measures, an assumption is made on the stationarity of the signal. This assumption is usually met in data collected in the field of behavioral neuroscience, in which many electrophysiological experiments are dominated by the repetition of equivalent (or considered equivalent) events. This type of data is non-stationary in its nature, but given the periodicity of the stimulus presentation (and the brain response), can be viewed as a cyclo-stationary process. However, in more recently emerging experimental paradigms, notably recordings acquired during "resting state" and in continuous brain computer interface (BCI) applications, no explicit cycling through event presentations and participant responses is featured. This type of data, again, is non-stationary, but may be modeled as a stationary dynamic process within a time window of a small enough size so the stationarity criteria is met (locally stationary).

We thus here propose and report initial results for two variants of an approach to estimating mutual information-based phase-amplitude coupling (MIPAC) in locally stationary and cyclo-stationary time series respectively. We will use the terms *Instantaneous MIPAC* for continuous data estimation and *Event-related MIPAC* for latency-tagged PAC estimation in multiple-trial event-related data.

The first steps for both procedures are to obtain appropriate phase φ_t and amplitude A_t time series from the data as in 2.3.1. At this point in the manuscript, we make a switch from the random variables X and Y used in the presentation of the local MI to the time series of the phase φ_t and amplitude A_t previously introduced in Section 2.3.1. Next, the local MI is estimated then as in Eq. 3.14 in a manner appropriate to the data structure.

Instantaneous MIPAC

To estimate *Instantaneous MIPAC*, we first compute the local MI between φ_t and A_t . For this, let us assume the space $Z = (\varphi_t, A_t)$. Then, for each point $z_i = (\varphi_i, A_i)$ in the Z -space, the number of neighbors $n_\varphi(i)$ and $n_A(i)$ are computed for a given value of k as done in Section 2.3.5. When estimating MIPAC, Eq. 2.18 can be rewritten as Eq. 2.21.

$$\|z - z'\|_z = \max \{ \|\varphi - \varphi'\|_c, \|A - A'\|_2 \} \quad (2.21)$$

Here, pairwise distances $\|\varphi - \varphi'\|_c$ and $\|A - A'\|_2$ are scaled by dividing each distance element by the maximum value of the pairwise distance matrix (alternatively, other type of scalings might be used with similar results, e.g. dividing by the spectral norm). Attending to the cyclic nature of φ_t , the circular ($\|\cdot\|_c$) norm [71] is used to compute pairwise distance between points in the phase space; the Euclidean ($\|\cdot\|_2$) norm is used for the amplitude space.

Then, to estimate the local MI between φ_t and A_t , the values of $n_\varphi(i)$ and $n_A(i)$ in the k -neighborhood are plugged into Eq. 3.14 as $n_x(i)$ and $n_y(i)$ respectively. The resultant local MI will be a vector of the same dimensions of φ_t and A_t .

To avoid the need of fixing a value for the parameter k , and as a convergence criteria for the estimation of the local MI, the local MI between φ_t and A_t is estimated in an iterative scheme for different values of k , starting at $k_0 = 1$ increasing by $\Delta k = 1$, until the percent decrease in local MI variance from step k_i to k_{i+1} (ΔVar_r) reaches the threshold ΔVar_{thres} , here defined as $\Delta Var_{thres} = 0.05\%$. The local MI for which the threshold ΔVar_{thres} is reached is then selected for further processing.

When computing Eq. 2.21, a leakage from phase frequency f_{phase} to $\|z - z'\|_z$ is observed. The characteristics of the leakage were determined empirically to lead to estimations of local MI with a high content of f_{phase} and its harmonics. As a heuristic to mitigate this leakage to obtain the *Instantaneous MIPAC* estimate, the estimated local MI

with the optimal k -value is low-pass filtered to below f_{phase} . Given the empirical nature of the joint space Z , a theoretical explanation of the high frequency leakage into the local MI may prove hard to obtain. However, the proposed heuristics should work independently of the nature of the leakage observed and is supported by the fact that we do not intend to interpret modulating frequency components above f_{phase} . The estimated *Instantaneous MIPAC* is represented by a vector with the same dimensions of φ_t and A_t .

Event-related MIPAC

In the *Event-related MIPAC* approach, similar to Voytek *et al.* [52] the low-frequency phase φ_t and high-frequency amplitude A_t time series are separated into non-overlapping epochs time-locked to a set of events of interest that are treated as being equivalent. Phase and amplitude epochs are then independently stacked to form two matrices (phase $\varphi_{trl,t}$ and amplitude $A_{trl,t}$) of dimensions number of trials by number of latencies. An across-trials local MI estimate between φ_t and A_t is then computed for each trial latency using the iterative scheme for estimating k presented in Section 2.3.6 on *Instantaneous MIPAC*. Notice here that estimated local MI at each latency will be a uni-dimensional vector of length the number of trials. To sample the full phase cycle on each latency neighborhood, the values of $n_\varphi(i)$ and $n_A(i)$ are estimated using a one-cycle f_{phase} frequency window centered on the latency of interest. This will also boost the nearest-neighbors count and account for possible jitter in the brain response to the stimulus. Estimation of local MI is then performed for each trial latency. Finally, to obtain the *Event-related MIPAC*, each epoch of the resulting local MI matrix (of length the number of latencies) is low-pass filtered to below f_{phase} to mitigate high frequency noise in the final MIPAC estimate.

Each resulting MIPAC epoch time series is interpreted as a time-varying measure of phase-amplitude coupling in the brain signal being analyzed. By taking the average of the *Event-related MIPAC* estimates across trials, we can obtain the *Trial-mean MIPAC*. Further collapsing the MIPAC time series by taking its grand mean gives a summary

Mutual Information Modulation Index (MIMI) value. This MIPAC-derived measure can be regarded as equivalent to other PAC estimates computed over trials and latencies (see Section 2.3.2).

2.3.7 Surrogate analysis to compute MIPAC significance

For all the methods described in Section 2.3.2 including both MIPAC variants, a statistical significance value can be computed using a surrogate data approach [27; 28].

Here, for each PAC estimate, $N_{surr} = 500$ surrogate measures were generated (unless otherwise specified). In the case of *Instantaneous MIPAC*, to obtain these surrogates, each φ_t and A_t time series were separated into $N_{segm} = 20$ (unless otherwise specified) segments. Segments in φ_t and A_t were then uniformly randomly shuffled across latencies and a new PAC measure estimate was derived. This process was repeated N_{surr} times to obtain a distribution of the surrogate MIPAC estimates. This shuffling process aims to disrupt any meaningful relationship between φ_t and A_t , while preserving some of the temporal structure of the original signal, to allow testing the null hypothesis that no meaningful coupling exists in the data. This procedure has been assumed to be a more rigorous test than random latency-shuffling [51; 72].

For *Event-related MIPAC*, surrogate signals for $\varphi_{trl,t}$ and $A_{trl,t}$ are constructed by randomly drawing as many time points as trials from a baseline of the respective matrices $\varphi_{trl,t}$ and $A_{trl,t}$. In this case, the baseline was assumed to be a latency window across all trials spanning from the beginning of the trials until the latency of the stimulus presentation.

For each surrogate data set we then compute the mean μ_{surr} and standard deviation σ_{surr} , plus MIPAC z -scores defined as:

$$Z_{surr} = \frac{PAC_{measure} - \mu_{surr}}{\sigma_{surr}} \quad (2.22)$$

Here, Z_{surr} values satisfying $|Z_{surr}| > 1.96$ were considered statistically significant for $\alpha = 0.05$ (assuming the z -scores to be gaussian distributed).

2.4 Results

Next, we present the results obtained to validate the *Instantaneous MIPAC* and *Event-related MIPAC* approaches introduced in Section 2.3.6, first using simulated and then using actual brain data.

2.4.1 Simulation results

As an initial validation of the MIPAC approach, we carried out five simulations. In the first two (Section 2.4.1 and 2.4.1) the performance of the *Instantaneous MIPAC* estimator was tested on simulated phase-amplitude coupled signals with and without added noise. In Section 2.4.1, we evaluate the performance of *Event-related MIPAC* and compare its estimates to those produced by the ERPAC measure (see Section 2.3.2). Finally, we compare the performance of MIMI (see Section 2.3.6) on a simulated PAC signal to the results of the PAC estimation methods reviewed in Section 2.3.2.

Estimating MIPAC on simulated single-trial PAC data

Here we study the estimation of MIPAC in a simulated continuous noiseless signal using *Instantaneous MIPAC*. To simulate the PAC signal, a synthetic Amplitude-Modulated (AM) signal was generated using a time-varying modulation strength. The modulation strength is then estimated using *Instantaneous MIPAC* and later is compared to the ground truth.

To generate the synthetic PAC signal, a procedure similar to that used for modeling AM signals was used (see A.1). Here we adopt the notation used in the AM literature. The signal and frequency of the modulator (f_m) in these simulations will correspond to the time series φ_t and the central frequency f_{phase} of the phase frequency band during

MIPAC estimation respectively. In the same way, the signal and frequency of the carrier (f_c) will correspond to the time series A_t and frequency f_{amp} of the amplitude frequency band used in the estimation.

In all five simulations, two sinusoids featuring frequencies $f_c = 40$ Hz and $f_m = 5$ Hz were generated to act as carrier and information (modulator) signals respectively. The duration of the simulated signals was five seconds at the sampling rate $S_{rate} = 500$ Hz. The signals were divided into five equal segments; coupling was introduced in Segments 2 and 4 only, leaving Segments 1, 3 and 5 with no coupling. In one simulation coupling was introduced in all segments. The threshold for variance reduction (see Section 2.3.6) was set to $\Delta Var_{thres} = 0.05\%$. Finally the signal was zero-padded by appending 1 sec of 0's to the beginning and to the end of each time series. To compute MIPAC while accounting for filter edge effects, this padding was removed after band pass filtering the signal. Unless otherwise specified, these parameters were used in all the simulations.

To assess the ability of *Instantaneous MIPAC* to retrieve the time course of coupling, three simulated variations in coupling strength were constructed without adding noise. On/off boxcar and linearly increasing time courses were used in the first two cases to simulate coupling in signal Segments 2 and 4 only. In the third case, coupling was applied as the absolute value of a sinusoid across the five Segments.

Fig. 2.2 shows the estimated time course of MIPAC in a simulated PAC signal in which modulation occurs only from 1 to 2 seconds and from 3 to 4 seconds; no coupling is induced outside of these data segments (Fig.2.2A). To estimate MIPAC, the simulated signal was band-pass filtered using a combination of high-pass and then low-pass filtering to isolate the two narrow frequency bands used in the simulation ($f_{amp} = f_c = 40$ Hz and $f_{phase} = f_m = 5$ Hz) as described in Section 2.3.1. The frequency limits of the band-pass filters were $f_{PhaseBand} = [4\ 6]$ Hz and $f_{AmpBand} = [34\ 46]$ Hz. These values were kept constant across simulations (any changes noted below).

Fig. 2.2B depicts the relationship between the (red) time series φ_t and (blue) A_t

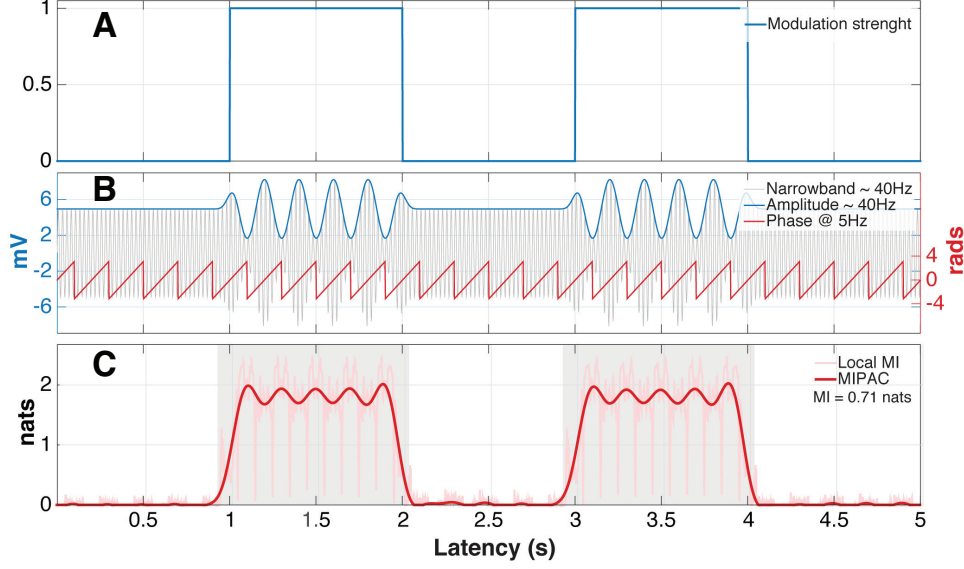


Figure 2.2. *MIPAC from a simulated PAC signal with on/off boxcar coupling.* Here the sampling rate of the simulated signal was $S_{rate} = 500$ Hz, the frequency of the carrier was $f_c = 40$ Hz and the frequency of the modulator was $f_m = 5$ Hz. (A): Coupling strength alternated between ‘on’ and ‘off’ across signal segments. (B): In the simulated signal: instantaneous low-frequency phase at 5 Hz (red), narrow-band high-frequency signal at 40 Hz (grey), and instantaneous high-frequency amplitude at 40 Hz (blue). (C): MIPAC estimated from the simulated signal for $f_{amp} = 40$ Hz, $f_{phase} = 5$ Hz. Time courses of estimated MIPAC (red) and local MI (only positive values are shown) before low-pass filtering to obtain the MIPAC (pink) are shown; both have the same units (nats). Overall MI between instantaneous phase and amplitude computed as the average local MI is shown in the legend. Latencies with statistically significant coupling estimates ($p < 0.05$, uncorrected) appear in light grey.

in this simulation. The time series A_t is the envelope of the (grey) narrow-band AM or carrier signal within $f_{AmpBand}$. The frequency of the carrier signal remains constant both during and in the absence of coupling, but its amplitude varies as a function of the phase of the modulator signal φ_t in the segments where the modulation is in effect.

Fig. 2.2C shows the MIPAC estimate (red) using the *Instantaneous MIPAC* approach along with the local MI (pink) time series obtained before being low-pass filtered to obtain the MIPAC. The overall MI computed as the average of the local MI is shown as well here. Note that MIPAC successfully captures the simulated modulation dynamics shown in Fig. 2.2A.

When MIPAC methods were introduced in Section 2.3.6, we discussed the high-frequency leakage in the local MI while estimating the k -neighbors in Eq. 2.21. Fig. 2.2C shows the effect of this leakage through the presence of a 10-Hz oscillation in the local MI. This frequency is the first harmonic of the f_{phase} frequency. The figure also shows how the heuristic remedy we implemented – low pass filtering the local MI at f_{phase} – improves the MIPAC estimate considerably. To this end, a zero phase-lag, infinite impulse response (IIR) Butterworth low pass filter [order 6] with a normalized cutoff frequency of $2 * f_{phase}/S_{rate} = 0.02$ was designed and implemented in MATLAB. The type of filter and filter order was chosen to avoid noticeable edge effects in view of the signal in question.

In performing the analyses to obtain the results shown in Figs. 2.3 and 2.4 we followed the same procedures as in Fig. 2.2 but used different time courses to vary the coupling: linearly increasing and absolute value of a sinusoidally varying coupling strength respectively. As can be seen in both cases, the MIPAC estimate successfully recovers the simulated variations in coupling strength.

Estimating MIPAC in simulated single-trial PAC data with added noise

In the previous section (Section 2.4.1), a set of three noiseless simulated PAC signals were used to test *Instantaneous MIPAC*. A similar MIPAC analysis was performed after adding white Gaussian noise to the same three simulated PAC signals using the MATLAB function *awgn.m*. For each signal, the signal-to-noise ratio (SNR) was set to 10. In Fig. 2.5, the local MI and MIPAC estimated from the noisy signal using the *Instantaneous MIPAC* approach are shown for simulated coupling time courses featuring a (on/off) boxcar, a linear ramp and absolute value of a sinusoid (Fig. 2.5A-C). The overall variations in PAC shown in Figs. 2.2A, 2.3A and 2.4A are again captured by MIPAC dynamics.

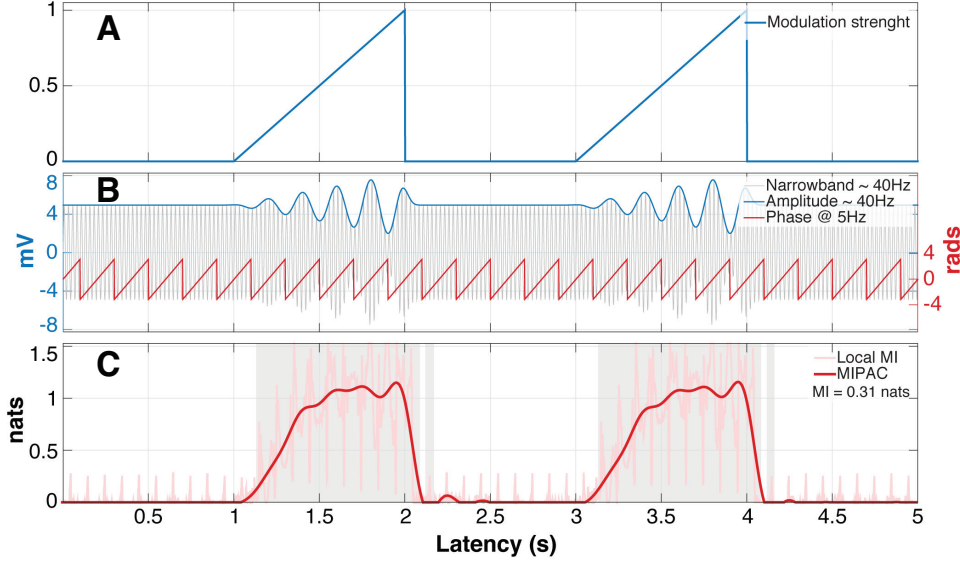


Figure 2.3. *MIPAC estimates of simulated PAC signals with linearly increasing coupling.* (A): Linearly increasing ramps of coupling strength in a simulated PAC signal. (B): In the simulated signal, instantaneous phase (red) at 5 Hz, (grey) narrow-band signal at 40 Hz and (blue) instantaneous 40-Hz amplitude. (C): MIPAC estimate for the simulated signal $f_c = 40$ Hz, $f_m = 5$ Hz. The (red) MIPAC estimate and (pink) local MI (only positive values are shown) before being low-pass filtered. Overall MI between instantaneous phase and amplitude computed as the average local MI is shown in the legend. Latencies with statistically significant coupling estimates ($p < 0.05$, uncorrected) appear in light grey.

Convergence in MIPAC estimation

The first row of Fig. 2.6 shows the percent decrease in the estimated local MI variance ΔVar_r (blue) and MIPAC variance (red) as a function of the number of k -neighbors in the simulations without noise added (Figs. 2.2-2.4). The maximum value of k on the abscissa represents the value at which the convergence criteria was reached for $\Delta Var_{thres} = 0.05\%$ (see Section 2.3.6). In this simulation, MIPAC variance tended to decrease as the value of k increased. The same trend, although less evident, is shown by ΔVar_r .

This result is intuitively correct. As k increases while keeping constant the number N of points in the signal, more neighboring time points are included in the estimation of the local MI for each specific point. This will lead to uniformity in the number of

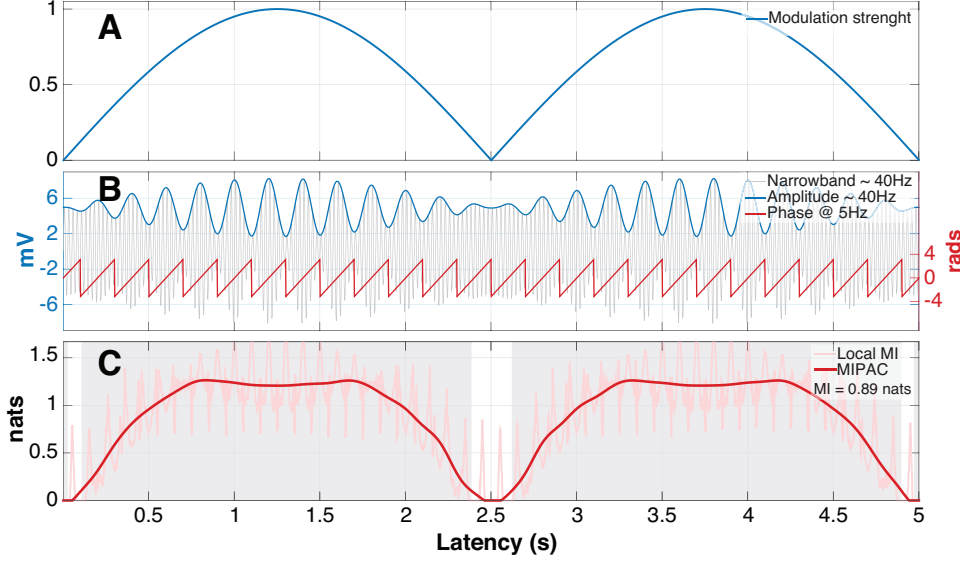


Figure 2.4. MIPAC time course estimate for a simulated PAC signal with absolute value of a sinusoid used as coupling. (A): Absolute value of a sinusoid used as coupling strength in a simulated PAC signal. (B): In the simulated signal, instantaneous phase (red) at 5 Hz, (grey) narrow-band signal at 40 Hz and (blue) instantaneous amplitude at 40 Hz. (C): Time course of MIPAC estimated from the simulated signal with $f_c = 40$ Hz, $f_m = 5$ Hz. The (red) MIPAC estimate and (pink) local MI (only positive values are shown) before being low-pass filtered. Overall MI between instantaneous phase and amplitude computed as the average local MI is shown in the legend. Latencies with statistically significant coupling estimates ($p < 0.05$, uncorrected) appear in light grey

neighbors for all points, thus pushing the distribution of local MI values toward uniformity and therefore decreasing the variance of the estimated MIPAC signal. Notice that in the limiting case $k = N - 1$ the number of neighbors for each point to be estimated become equal, which lead to $\Delta Var_r = 0$ and MIPAC variance = 0.

The same convergence analysis performed on three noiseless signals was replicated for the noisy simulated signals ($SNR = 10$) (lower row of Fig. 2.6). The pattern of behavior previously found in the noiseless simulations was preserved, though in this case as k increased the decreases in ΔVar_r and MIPAC variance became smoother. This effect can be explained by the fact that by introducing noise, the distribution of points z_i in the joint space generated by the supports of A_t and φ_t become more uniformly distributed. This blurs the signatures of interdependence between the signals in the Z -space and leads

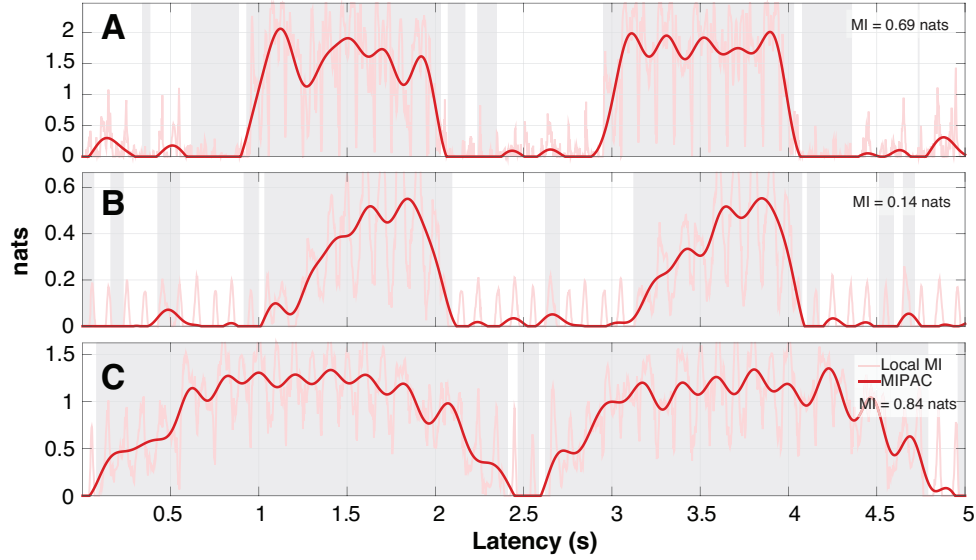


Figure 2.5. *MIPAC estimation using Instantaneous MIPAC on simulated PAC signals with noise added.* In this simulation, the time course of MIPAC was estimated for the same signals simulated in Figs.2.2-2.4 but with noise added ($SNR = 10$). Estimated (red) MIPAC, and (pink) local MI before low-pass filtering are shown. Latencies with statistically significant coupling estimates ($p < 0.05$, uncorrected) appear in light grey.

to a smoother MIPAC estimate. Notice that in the limiting case when $SNR \rightarrow 0$, the plots in second row of Fig. 2.6 will display an approximate similar $\frac{1}{k}$ decay.

Another simulation was performed to study interactions between the sampling rate, S_{rate} , the SNR of the signal, and the parameter k used in MIPAC estimation. Simulated PAC signals of five seconds duration were generated using all combinations of the parameters $SNR = [0.2, 0.5, 1.4, 3.8, 10]$ dB and $S_{rate} = [128, 256, 512, 1024, 2048]$ Hz. Coupling was introduced as in Fig. 2.2. *Instantaneous MIPAC* in these signals was then computed using a fixed value of $k = [2, 3, 4, 6, 8, 12, 17, 24, 35, 50]$. For each combination of these parameters, the estimated MIPAC time series and the time course of simulated coupling were correlated. The values of these correlations for all combinations of parameters S_{rate} , SNR and k are shown in Fig. 2.7. Here, higher correlation values reflect a better MIPAC estimation of the simulated coupling.

Fig. 2.7 shows that MIPAC estimation is negatively impacted by higher noise levels

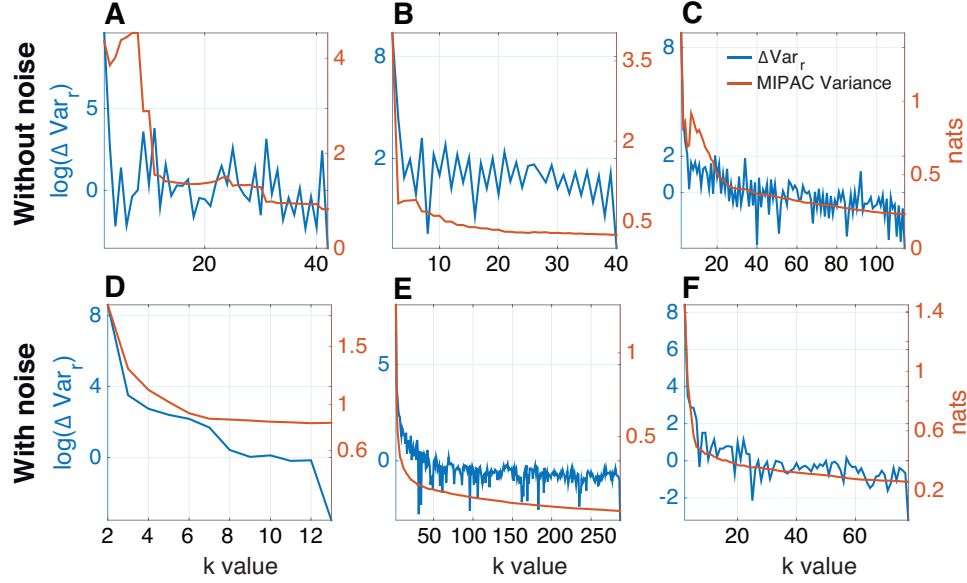


Figure 2.6. *Convergence in MIPAC estimation.* Percentage variance reduction (ΔVar_r) (blue) and MIPAC variance (red) as a function of the number of k -neighbors. The decreasing of values of ΔVar_r below the threshold defined by ΔVar_{thresh} defines the convergence criteria for the MIPAC estimate. The three columns in the figure show the percentage variance reduction (ΔVar_r) (blue) and MIPAC variance for each of the three types of simulations shown earlier without noise (A-C) and with noise added (D-F)

but that higher S_{rate} improves MIPAC reliability. The dependence of the MIPAC estimate on k was not as strong as its dependence on S_{rate} . For values of k below 4 and above 24, MIPAC estimation performance was poor, particularly at lower SNR levels. Values between $k = 6$ and $k = 12$ appear to provide slightly better estimates for a given SNR . The effect of parameter k seems to affect the MIPAC estimate only for low values of SNR and S_{rate} .

Estimating MIPAC on simulated multi-trial data

Next, we tested the *Event-related MIPAC* approach, estimating MIPAC for a simulated multi-trial data set, and compared the results to the PAC time series computed for these data using ERPAC [52]. For this simulation, a synthetic PAC signal similar to that shown in Fig. 2.2 was first used. The signal was replicated $N_{trials} = 200$ times and stacked to form a matrix of dimension N_{trials} by number of latencies. Then, each trial was

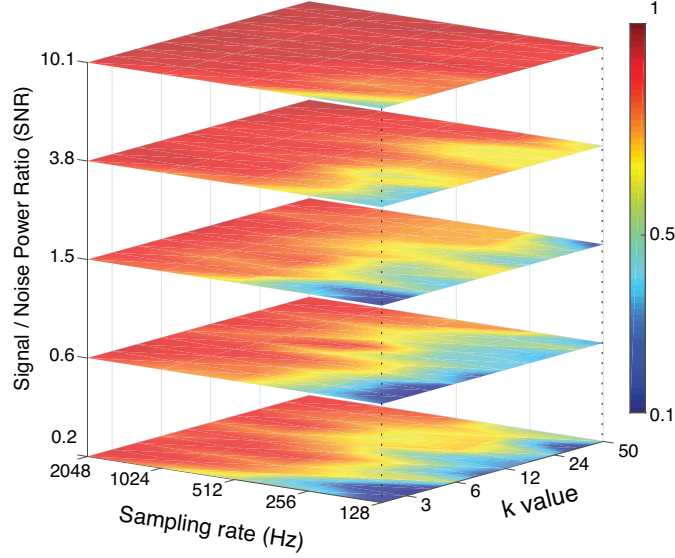


Figure 2.7. *Relationship between MIPAC estimates and signal parameters.* Functional relationships between MIPAC estimates and parameters k , S_{rate} and SNR were assessed by computing the correlation of the estimated coupling time courses and *Instantaneous MIPAC*.

circularly shifted a random number of points between 1 and 100 (generated from a uniform distribution), equivalent to one full cycle of the simulated phase frequency. This was aiming to simulate actual trial-to-trial variations of brain responses to a repeated and/or equivalent stimulus presentation. Gaussian white noise was then added to each trial, with $SNR = 10$. Phase and amplitude time series for each trial were estimated as described in Section 2.3.6. Then *Event-related MIPAC* was computed and its mean progression across the trials (*Trial-mean MIPAC*) calculated. The MIPAC time series estimate for each trial is shown in the central panel of Fig. 2.8; The *Trial-mean MIPAC* time series is shown in the lower panel. *Trial-mean MIPAC* (red trace) and the time-resolved ERPAC (blue trace) are similar (Fig. 2.8, bottom).

Mutual Information Modulation Index (MI_{mi})

Next, we compared the performance of the MI_{mi} (see Section 2.3.6) against the modulation indices obtained by using MVL_{mi} (Section 2.3.2), KL_{mi} (Section 2.3.2) and

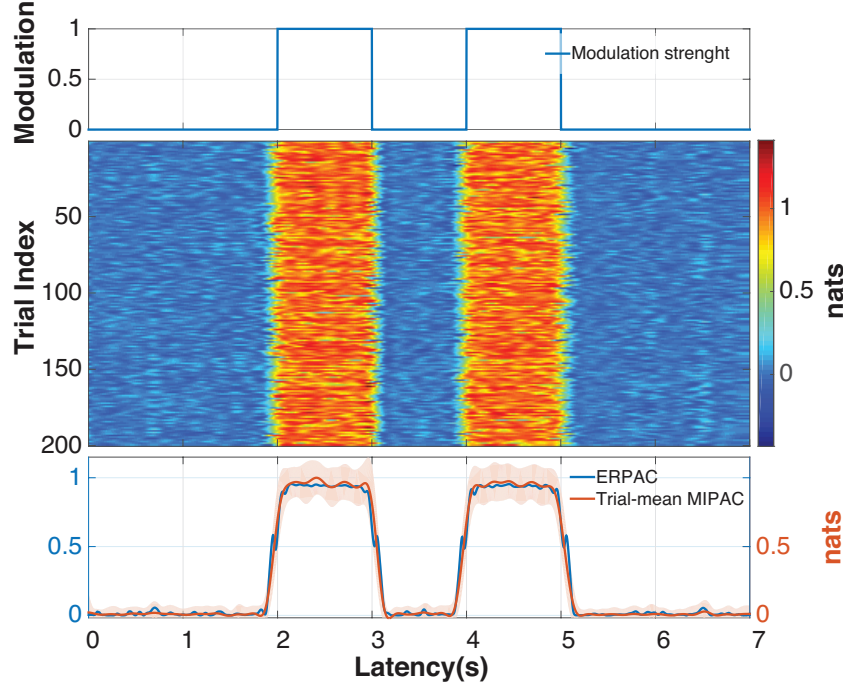


Figure 2.8. *Estimating MIPAC in a simulated noisy multi-trial PAC data set.* A set of 200 simulated PAC signal trials like that depicted in Fig. 2.2 were generated to estimate *Event-related MIPAC*. Each trial was circularly shifted a random number of points (between 1 and 100), and Gaussian white noise was added to each trial, setting the $SNR = 10$. *Top:* An on/off boxcar waveform was used to vary the modulation strength of the simulated PAC signals. *Middle:* Estimated MIPAC for each trial and latency. *Bottom:* Estimated ERPAC (blue) and *Trial-mean MIPAC* (red). Standard deviation of MIPAC across trials is shown in light red.

GLMmi (Section 2.3.2). We first simulated a signal similar to that shown in Fig. 2.2, with an (on/off) boxcar waveform modulating the coupling and using $f_c = 50$ Hz, $f_m = 7$ Hz. White Gaussian noise was then added to make $SNR = 10$. The simulated signal is shown in Fig. 2.9A. Modulation Index was computed on this signal for each of the combinations of $f_{phase} = [4, 5, 6, 7, 8, 9, 10, 11, 12]$ Hz and $f_{amp} = [30, 35, 40, 45, 50, 55, 60, 65, 70]$ Hz by using MVLmi, KLmi and GLMmi. Additionally *Instantaneous MIPAC* was computed for all combinations of f_{phase} and f_{amp} (Fig. 2.9B). MImi image (Fig. 2.9C) was then computed by taking the mean of MIPAC time series. MImi image was then compared with GLMmi (Fig. 2.9D), MVLmi (Fig. 2.9E) and KLmi (Fig. 2.9F) modulation index estimates. To facilitate comparison across methods, modulation indices were scaled by

dividing the values on each figure by its maximum modulation value. The MI_{mi} estimates qualitatively replicate the results obtained using the other methods and even outperform the KL_{mi} (Fig. 2.9F).

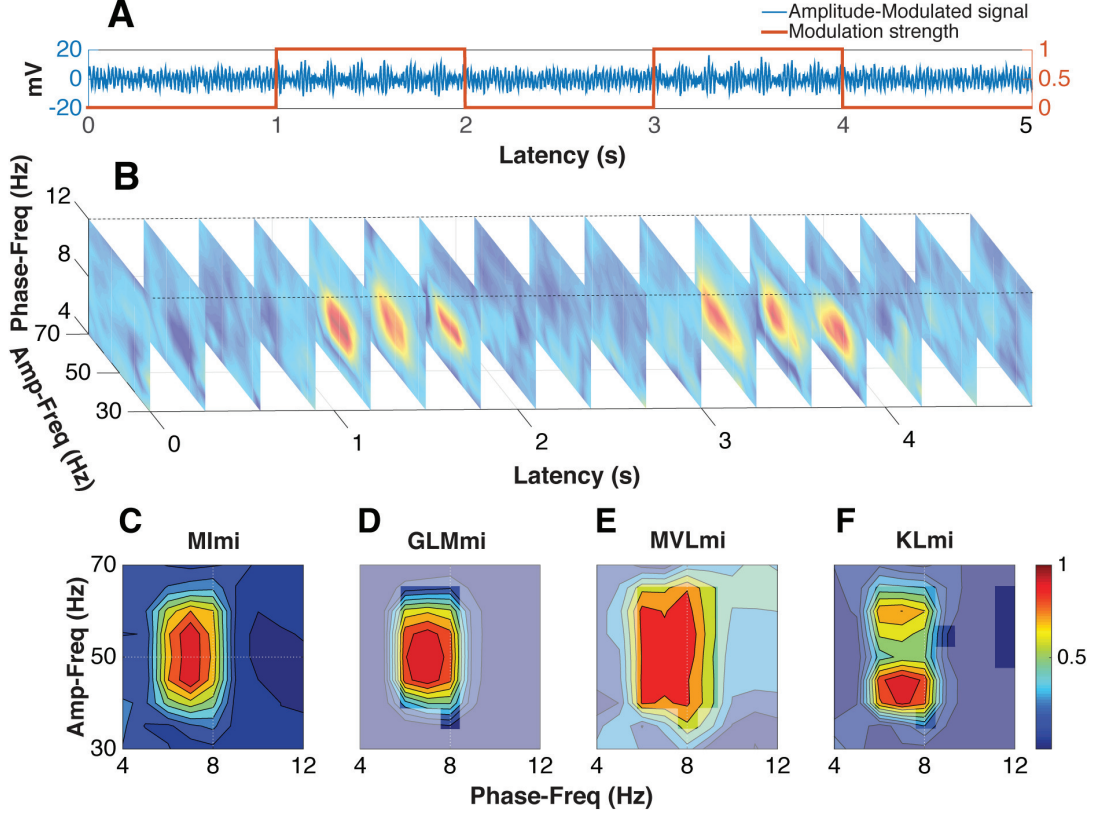


Figure 2.9. Comparing results of MI_{mi} with those of other modulation indices (A): Simulated PAC signal with noise added ($SNR = 10$) (blue). An on/off boxcar waveform is used to model the PAC modulation strength (red) in the simulated signal. In this simulation, the amplitude frequency is $f_c = 50$ Hz, the phase frequency is $f_m = 7$ Hz, and the sampling rate is $S_{rate} = 500$ Hz. (B): MIPAC estimates for all combinations of phase frequencies from 4 to 14 Hz (1 Hz steps) and amplitude frequencies from 30 to 70 Hz (5-Hz steps). Comodulograms using (C) MI_{mi} , (D) GLM_{mi} [27], (E) MVL_{mi} [28] and (F) KL_{mi} [26]. Non significant values ($p < 0.05$, uncorrected) appear shaded in (D-F).

2.4.2 Applying MIPAC to actual ECoG data

Subject selection and data exploration

We applied *Event-related MIPAC* on actual electrocorticographic (ECoG) data from a human subject. Results obtained in this section are aiming to illustrate the use of the proposed method with actual data, not to support any specific interpretation of the results nor to generalize these results in advance to any other data sets.

The data described in Section 2.3.1, with electrode locations over the inferior temporal gyrus, fusiform gyrus, and lingual gyrus in the left brain hemisphere, were used after being preprocessed as described in Section 2.3.1. Combinations of f_{phase} , ranging from 2 Hz to 20 Hz in 2-Hz steps, and f_{amp} from 30 Hz to 120 Hz in 5-Hz steps, were investigated. The time series A_t and φ_t for all combinations of these frequency values were computed (as described in Section 2.3.1) and split into epochs time-locked to either 'face' image or 'house' image stimuli and extending from -400 ms to 800 ms relative to stimulus onset. Next, *Event-related MIPAC* was estimated across all ECoG channels separately for face and house stimulus-locked datasets. Grand mean estimated MIPAC across trials and latencies was computed for each frequency combination to obtain two comodulogram matrices of MImi estimate (Fig. 2.10).

A single channel exhibiting PAC was selected to illustrate the application of the proposed method to actual brain data. The selected channel (Channel 16) was located on the posterior side of the fusiform gyrus. The maximum MImi value was obtained for 16 Hz (phase) and 95 Hz (amplitude) (Fig. 2.10 upper left dotted area) following face presentations only (Fig. 2.10). The channel Event-Related Potentials (ERPs) time-locked to face and house image presentations are plotted in Fig. 2.11 (left and right panels). For the selected channel, the responses to face presentations are more consistent across trials than responses to house presentations. Mean responses to face stimuli (Fig. 2.11, lower left) show a distinctive evoked negative peak (N170) at roughly 180 ms after presentation,

while mean responses to house stimuli (Fig. 2.11, lower right) exhibit a weak positive peak near 420 ms after stimulus presentation. These ERP-image panels were plotted using EEGLAB function *erpimage.m* using a 3-trial vertical smoothing window.

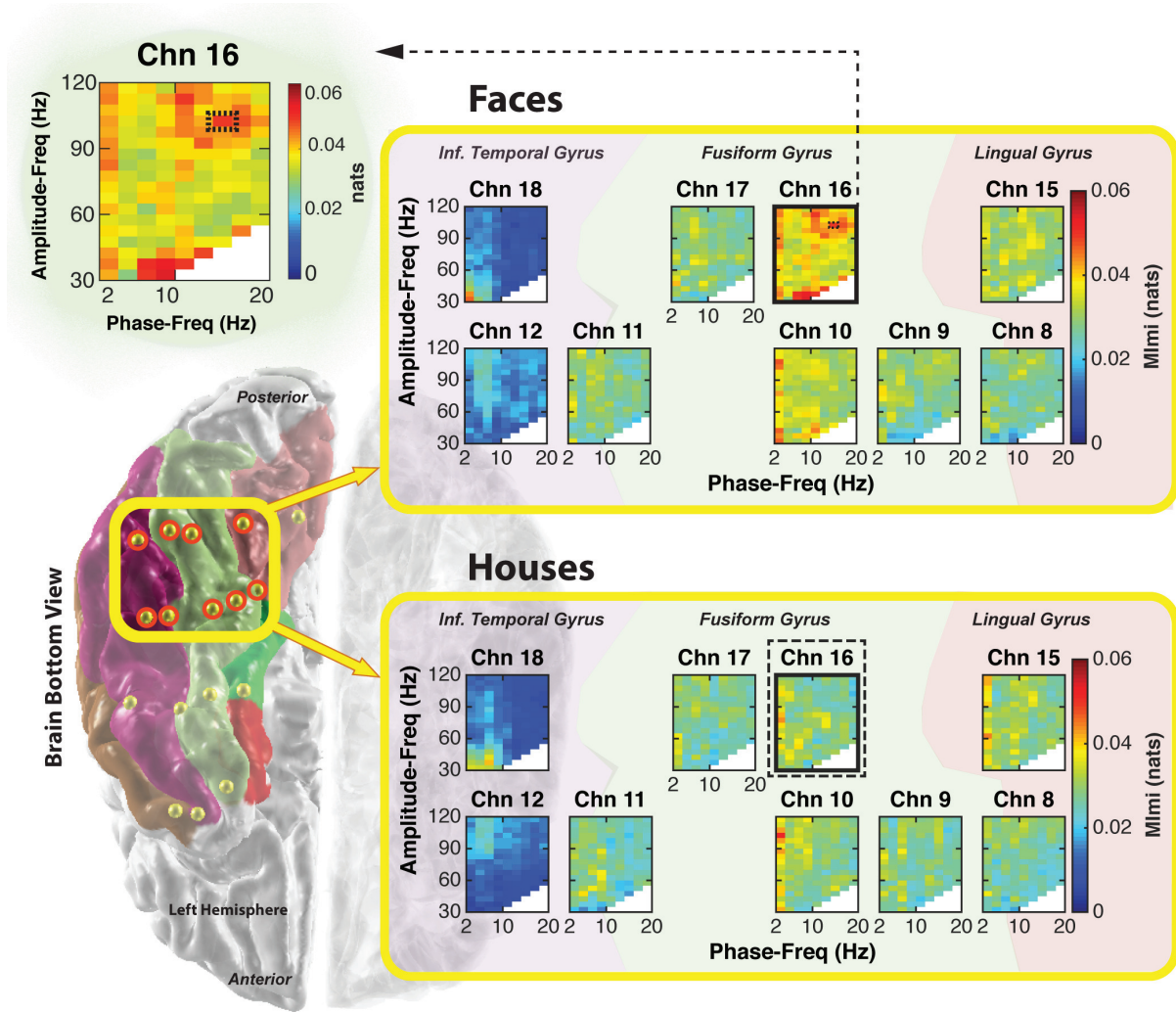


Figure 2.10. *Electrode locations (lower left) and MIMI comodulograms computed for face and house stimulus-locked trials at a subset of ECoG channels electrodes. The comodulogram for responses to face presentations in Channel 16 is magnified in upper left. The MIMI maximum (95-Hz amplitude, 16-Hz phase) is highlighted with a dotted box (upper left). Note that there is no MIMI following house image presentations for this channel (dashed box lower right). No significance testing was performed on the comodulograms.*

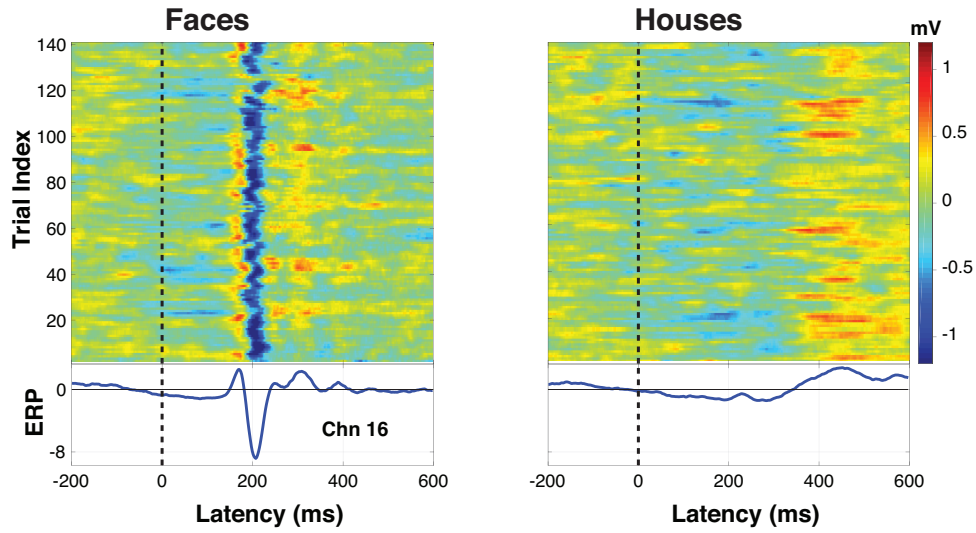


Figure 2.11. *ERP-images for Channel 16 trials time locked to presentations of face and house image stimuli.* Stimulus onset is at 0 ms and trials have been (vertically) smoothed with a 3-trial moving window. Left: Single ECoG trials time-locked to the presentation of face stimuli. Right: Trials time-locked to presentations of house stimuli. Lower panels plot the trial-mean ERP for each trial subset. At Channel 16, located on the Fusiform Gyrus, the response to face stimuli includes a prominent and expected 'N170' negativity that does not appear following presentations of house stimuli.

To study the spectral dynamics of brain responses to presentations of face and house images, event-related spectral perturbation (ERSP) [73] and inter-trial coherence (ITC) measures were computed for the same channel (16). Results are shown in Fig. 2.12. In the ERSP time locked to face-stimulus presentations (Fig. 2.12, upper left panel), beginning near 150 ms after stimulus onset a sharp power increase occurs across a broad band spanning from 4 Hz to above 120 Hz. At lower frequencies (≤ 50 Hz) this power increase is brief, lasting ~ 100 ms. At higher frequencies, power decreases slowly from its peak near 150 ms. ITC time locked to face stimulus presentations (Fig. 2.12, lower left panel) becomes significant (≥ 0.4) in the (10-30 Hz) alpha-beta frequency range during the N170 negativity (between 150 ms and 250 ms). In ITC following house stimulus presentations (Fig. 2.12, lower right), a slight increase occurs near 400 ms at low frequencies (≤ 20 Hz) during an ERP positivity, a feature that does not appear in the house stimulus ERSP (Fig. 2.12, upper right).

Due to the non-stationarity in the signal indicated by the relatively sharp response to the face stimulus, it may be a concern that in the time-frequency analysis the power in high frequencies might be artificially created in response to sharp spike-like features in the (single-trial) signals. This issue is a well known and long discussed [74] side effect of all Fourier-like techniques that may lead to spurious PAC, since artificial high frequencies returned by the analysis may then be coupled in time to slower oscillations. While this can occur in some cases, in our case high-frequency components resembling wave complexes (≥ 50 Hz) can be seen in single responses to face image presentations. In Fig. 2.13 we show two such trials (trials 113 and 91 in Fig. 2.11) time-locked to face stimulus that display this effect. Notice here that the increase in the power of the gamma band around 95 Hz (Fig. 2.13 red traces) is coherent with the high-frequency oscillation superimposed to the ERP starting around 200 ms (Fig. 2.13 blue traces). This argues for the validity of the high-frequency components found in the data (Fig. 2.12), but nevertheless is an issue that may need special attention in the general case.

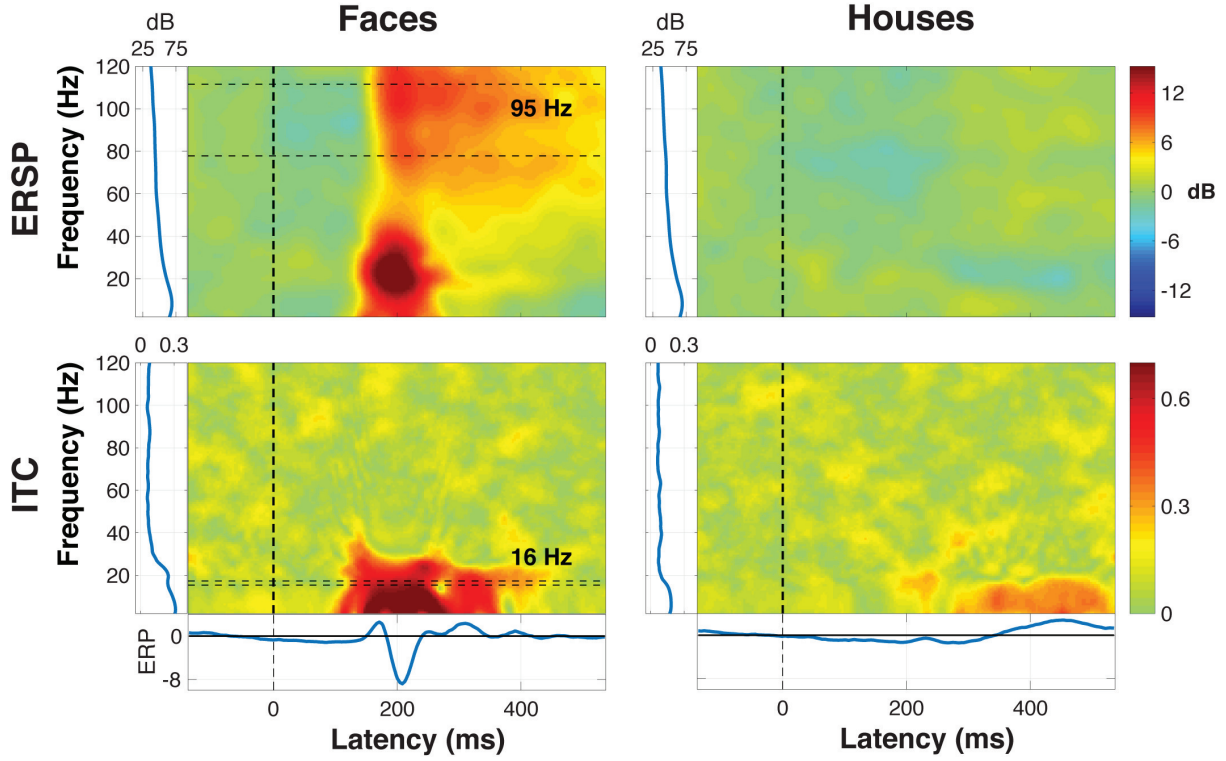


Figure 2.12. Event-related spectral perturbations (ERSPs) and inter-trial coherences (ITCs) for the same channel (16) time locked to (*left*) face and (*right*) house image presentations. The trial-mean ERPs are shown in the traces below the ITC panels. The face-stimulus ERSP exhibits a broadband power increase near 200 ms after stimulus onset. Beta-band frequencies (16-30 Hz) exhibit strong (≥ 0.5) ITC near 200 ms. The ERSP and ITC responses to house stimuli (*right*) are weaker and peak later (400 ms). Jointly, these phenomena may suggest the presence of event-related phase-amplitude coupling (PAC) near $f_{phase} = 16$ Hz and $f_{amp} = 95$ Hz. Attending to Section 2.3.1, these bands are represented with horizontal dotted lines with values between [78, 112] Hz (*upper left*) and [15, 17] Hz (*lower left*) respectively.

Trial-mean MIPAC and ERPAC

In the face-image trials ERSP shown in Fig. 2.12, the simultaneous event-related increases in power in the beta and gamma bands (maximal in broad sub-bands containing 95 Hz ([78, 112] Hz) and 16 Hz ([15, 17] Hz) sub-bands respectively) coincide with a peak in ITC at ~ 16 Hz surrounding the negative N170 trough. This indicates the presence of PAC at these frequencies. This finding is consistent with the relatively high degree of MImi found for this channel (Figure 10).

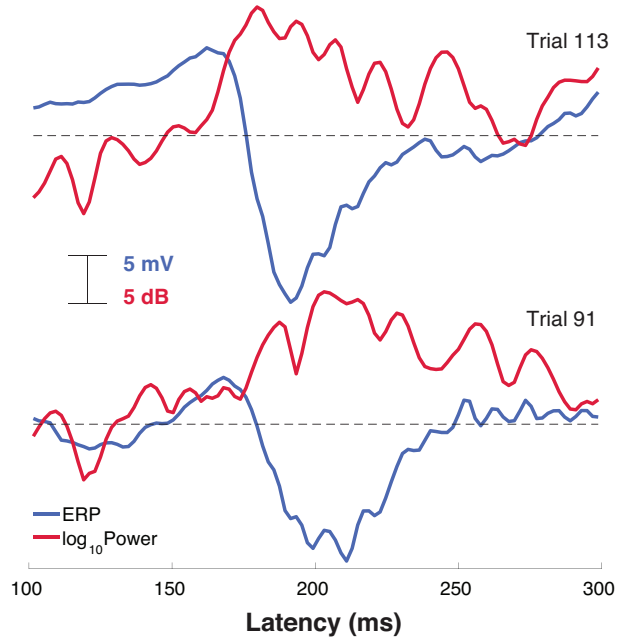


Figure 2.13. *Two single face image presentation trials displaying high-frequency activity:* The figure shows (blue traces) trials 91 and 113 in Fig. 2.11 and time courses for the same two trials of power near 95 Hz (band limits, 78 Hz to 112 Hz). The computed power increases are consistent with the appearance of high-frequency oscillations superimposed in the ERPs surrounding the negative (‘N170’) ERP peak

Next, *Event-related MIPAC*, *Trial-mean MIPAC* and ERPAC were computed for face and house image trials. Results of *Trial-mean MIPAC* and ERPAC were compared (Fig. 2.14). *Trial-mean MIPAC* for both stimulus trial sets (Fig. 2.14, upper panel) is consistent with the results from ERPAC (Fig. 2.14, lower panel). Statistical significance of increases in *Trial-mean MIPAC* and ERPAC following face-image presentations (compared to pre-stimulus baseline) were computed using a family-wise-error rate (FWER), t_{max} permutation test [75], and are shown in Fig. 2.14 in pink shading. The empirical distribution of maximal t -values under the null hypothesis was obtained by generating 5000 100-time point surrogate epochs from the pre-stimulus baseline. The null hypothesis was then rejected at a significance level $\alpha = 0.05$ if the actual statistic t of the analyzed sample was beyond the $1 - \alpha$ percentile of the empirical distribution. No significant differences from baseline were found in responses to house image presentations. Thus, further analyses

focused on responses to face image presentations.

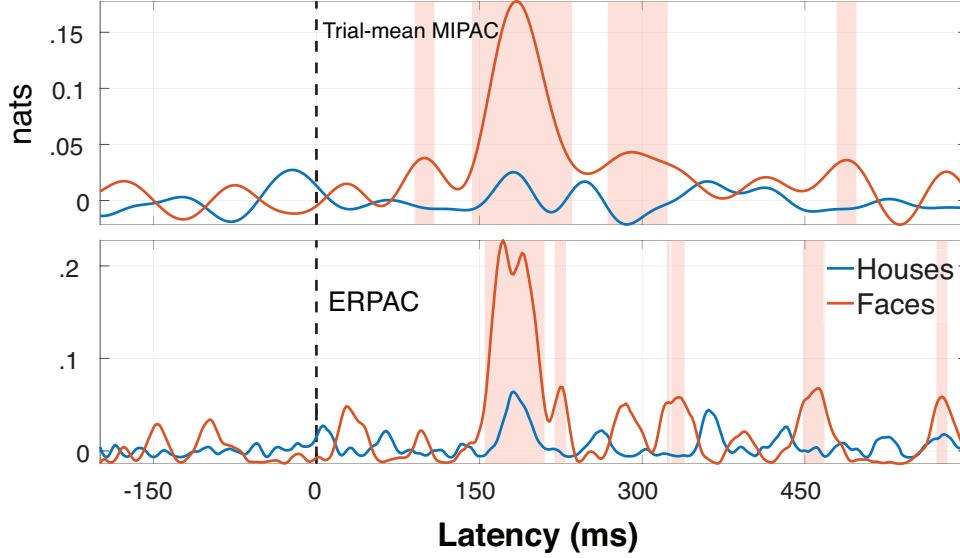


Figure 2.14. *Trial-mean MIPAC and ERPAC computed for Channel 16 at f_{phase} 16 Hz and f_{amp} 95 Hz. Top: Trial-mean MIPAC following face image (red) and house image (blue) presentations. Bottom: ERPAC following face image (red) and house image (blue) presentations. Pink shaded areas indicate intervals of significant PAC ($p < 0.05$, FWER corrected over both time and conditions) following face image presentations. Both event-related MIPAC and ERPAC measures give similar results that feature a PAC maximum near 185 ms following face image presentations only.*

Event-related MIPAC

Results of MIPAC estimation using *Event-related MIPAC* ($f_{phase} = 16$ and $f_{amp} = 95$ Hz) for all face-presentation trials are shown in Fig. 2.15A, which we may refer to as a (peak MIPAC-sorted) MIPAC-image. In this figure, a statistical significance mask ($p < 0.05$, uncorrected) was computed by surrogate data analysis (see Section 2.3.7); non-significant values were set to zero and are shown in green color. In the MIPAC-image, trials were sorted by mean MIPAC across a window centered at 200 ms and spanning the equivalent of one full cycle of f_{phase} (the region between the dotted lines). In this time window, MIPAC consistently increases from bottom to top across the (MIPAC-sorted) trials. The same trial sorting order was then used to plot the time course of instantaneous phase at 16 Hz (Fig. 2.15C) and power at 95 Hz (Fig. 2.15B) in the same trials. The

interval of increased MIPAC (180 to 250 ms) co-occurs with the N170 ERP peak (see Fig. 2.15D). However the MIPAC values are not homogeneous across trials. Trials with higher MIPAC in the sorting window also exhibit a distinct power increase at 95 Hz (Fig. 2.15B). This occurs concurrently with a partial phase reset at 16 Hz (Fig. 2.15C) beginning just after stimulus presentation and lasting until about 430 ms as also indicated by the ITC image. Here, latencies showing negative MIPAC may be interpreted as exhibiting no coupling.

The Fig. 2.15G shows the time course of 16-Hz phase in face image trials now explicitly sorted by phase at 16 Hz in a window centered at 200 ms spanning one full cycle at f_{phase} (e.g., between the thin dotted vertical lines). Using this trial-sorting order, the phase resetting after the stimulus presentation becomes even more obvious (Fig. 2.15G versus C). The same trial-sorting order was used then to plot the MIPAC-image (Fig. 2.15E) and the time course of power near 95 Hz in each trial (Fig. 2.15F). For the MIPAC-image (E), as in A, a statistical significance mask ($p < 0.05$, uncorrected) was computed using surrogate data analysis. We can see that the three measures depicted here, MIPAC (E), high-frequency amplitude (F), and low-frequency (16-Hz) phase (G), all appear to co-vary in latency across trials during the N170 negative peak (H).

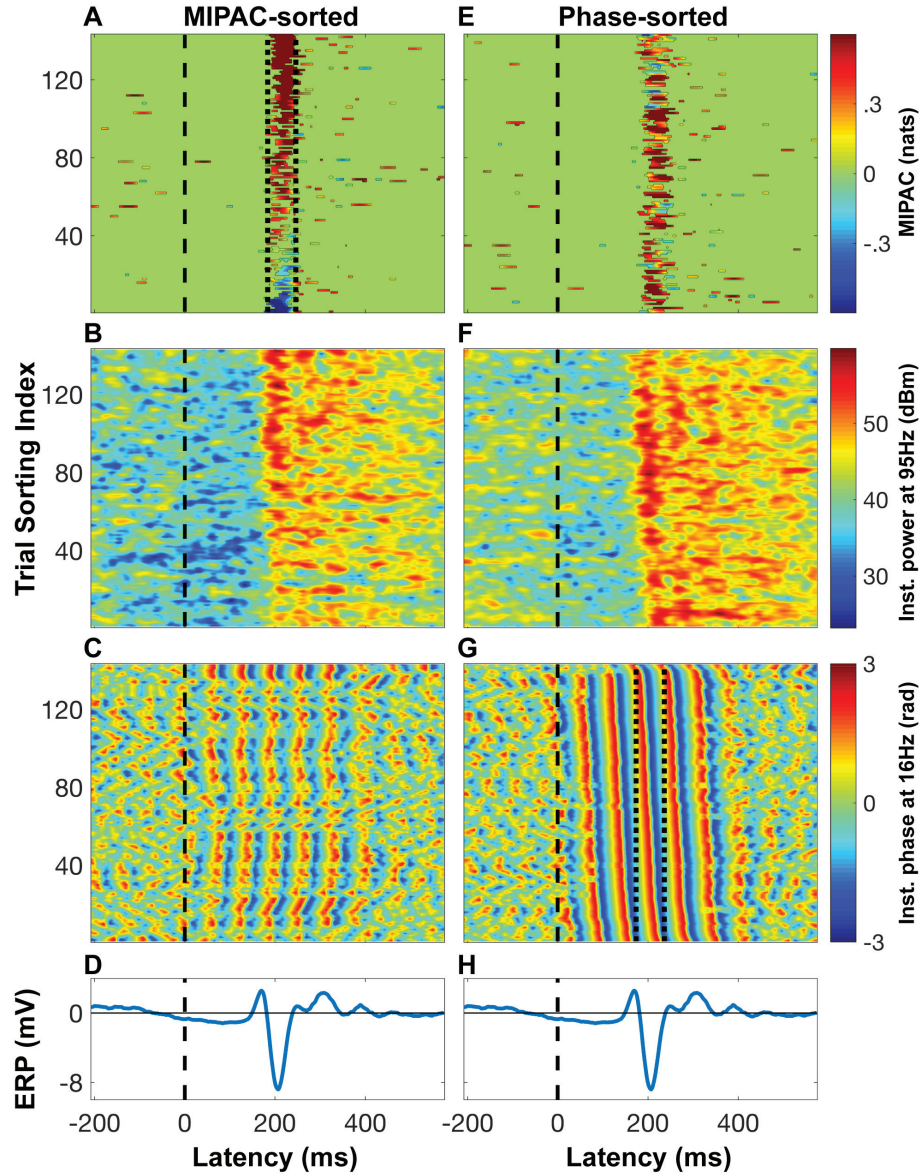


Figure 2.15. *MIPAC-images showing event-related MIPAC time series for individual face image trials.* (A): MIPAC-image with trials sorted by the mean MIPAC value near 200 ms (between the vertical dotted lines). Points with statistically non-significant MIPAC estimates ($p < 0.05$, uncorrected) were set to zero and are shown in green color. (B): Power near 95 Hz in trials sorted as in A. (C): Phase at 16 Hz, trials again sorted as in A. (D): Trial-mean ERPs. (E-H) as in A-D, trials here sorted by 16-Hz phase in a latency interval centered at 200 ms (see dotted lines in panel G). Images in panels A and E are smoothed with a 3-trial moving window.

2.5 Discussion

2.5.1 MIPAC estimates of simulated data

We showed that both MIPAC variants, *Instantaneous MIPAC* and *Event-related MIPAC*, are able to successfully estimate the temporal dynamics of the coupling in simulated PAC signals both without and with added noise. In simulations, as the noise level in the simulated signal increased, MIPAC estimation accuracy decreased. Although analyzed here only in three comparable simulations, we expect this result to generalize. Noise introduced into the simulated signals produces an increased uniform spread of points in the joint space spanned by the supports of A_t and φ_t , leading to more uniformity in the numbers of neighbors counts used in the local MI computation and undermining the estimation of MIPAC. A quantitative evidence for this is the decrease of the overall MI values from the noiseless simulations in Fig. 2.2, 2.3, 2.4-C from 0.71, 0.31 and 0.89 nats to 0.69, 0.14 and 0.84 nats respectively in the simulations with noise added in Fig 2.5A-C.

We also studied the reliability of estimating coupling dynamics using *Instantaneous MIPAC* in single trials. For this, we assessed the relationship of the signal parameters SNR and S_{rate} to the number of neighbors k used in MIPAC estimation (Fig. 2.7). Correlation between the modeled coupling strength and estimated MIPAC time series was used as a measure of the quality of MIPAC estimation under the given parameters. Similar to other PAC estimation methods [27], we showed that the reliability of the MIPAC estimation is inversely proportional to noise level and directly proportional to the signal sampling rate. In our method, the later correlation occurs since as the S_{rate} increases, the joint space spanned by the supports of A_t and φ_t becomes more locally populated, leading to increased estimates of marginal neighbor counts for points with relevant PAC. Here, with locally populated space, we want to highlight the difference between the increase of uniformity introduced by the increase of the noise level and of the local density increase in the joint space evoked by the increase of S_{rate} in signals displaying PAC.

In the same simulation, for a given S_{rate} we were able to identify a k -value interval that gave a high-reliability MIPAC estimate. This result supports our conclusion that, given a signal with PAC (real or simulated), an optimal value of k can be estimated. This supports our implementation of a heuristic in the form of the iterative scheme for the k -value used to estimate MIPAC here. However, the optimal values of k obtained cannot be generalized, since the result is dependent on the dynamics of the coupling in each signal. As a heuristic, the proposed iterative (or convergence) scheme is not a robust method to uniquely determine the MIPAC estimate, but rather gives a coarse approximation of the optimal k . Indeed, in our examples the combination of this heuristic and the low-pass filtering at the end of MIPAC estimation produced fair estimates. We also note that small values of the parameter Var_{thres} may lead to the limiting case described in Section 2.4.1 where the estimated coupling can become constant, or analogously, have zero variance. As this is an undesirable feature, a careful assessment of parameter Var_{thres} is encouraged in real data applications. These results were obtained while estimating MIPAC in a single-trial signal using *Instantaneous MIPAC*. However, if we consider *Event-related MIPAC* as an iterative application of *Instantaneous MIPAC* at each latency across trials, then these results can be applied and generalized without loss of rigor to the case of multiple trials. For such data, *Event-related MIPAC* takes advantage of the presumed cyclostationarity of the event-related trials to boost the neighbor count in the MIPAC estimation. The use of the one phase-cycle estimation window both allows the MIPAC estimates to sample from all parts of the phase-frequency phase cycle, but also can better capture transient PAC across trials in which the event-related PAC time courses have some latency differences. On a practical level, these considerations support the reformulation of *Instantaneous MIPAC* into *Event-related MIPAC*.

In the simulation 2.4.1, MIPAC computed on a set of simulated PAC signals was compared to the PAC result obtained using ERPAC. We demonstrated that both ERPAC and *Trial-mean MIPAC* could successfully estimate changes in PAC modulation strength.

In addition, *Event-related MIPAC* estimated the modulation strength of PAC dynamics in each individual trial. From our view, this is one of the main benefits of the proposed MIPAC method vis a vis existing methods.

Finally, in Section 2.4.1 we compared MImi performance on a simulated signal to the modulation indices obtained by GLMmi, MVLmi and KLmi. Here we demonstrated the degree of agreement among the results across all these methods. In our work, MImi was introduced as an alternative to the modulation index used by other authors [26–28]. Our alternative came from the equivalence of MIPAC, when computed without low pass filtering of the local MI, to the overall MI. This measure has the potential to be used in analyses like the one performed at the beginning of Section 2.4.2, but note that this improvement may come at the cost of compromising critical temporal information about the PAC process.

2.5.2 MIPAC estimates of experimental data

To illustrate the potential of the proposed method, *Event-related MIPAC* was applied to real ECoG data in Section 2.4.2. Here a single, average-referenced ECoG channel with the active electrode located in the fusiform gyrus and exhibiting high MImi in response to face image presentations was selected for further investigation. Selecting a channel because of a particular feature in its comodulogram, then analyzing the very same feature on the channel, might constitute a classic example of *double dipping* [76]. Also, most of the presented statistical analysis lacks multiple comparison correction (e.g., over sensors, conditions, frequencies). However, as mentioned previously, this analysis was presented here for illustrative purposes, without investigating whether the results might or not generalize to new subjects, etc..

The specificity of the fusiform gyrus response to visual face image presentations, and its expected (N170) ERP response to face image presentations is well known [77]. However, the observed increase of beta-gamma band PAC during the surface-negative

N170 ERP peak is a novel observation that may deserve further investigation. Here, the high-frequency broadband (HfB) activity increase in the ERSP (Fig. 2.12) following face presentations is consistent with the phase-amplitude coupling increase. The simultaneous ~ 16 -Hz ITC and ~ 95 -Hz HfB power increases occur together during the PAC period. The increase in power near 16 Hz is a necessary condition for emergence of meaningful PAC, since filtering may create an artificial appearance of activity at the phase frequency, which can lead to a spurious correlation. Interestingly, the local PAC values across trials (Fig. 2.15A,E) are not homogeneous.

Fig. 2.14 shows that results of *Trial-mean MIPAC* are similar to those obtained using ERPAC on the same data. This supports the validity and potential utility of the new MIPAC measure. Here the *Trial-mean MIPAC* was computed from the *Event-related MIPAC* estimates only for comparison to another PAC estimation method (ERPAC). MIPAC offers a richer description of PAC dynamics than other methods by providing a time-resolved PAC estimate for each trial.

Here, whether or not PAC is a brain mechanism supporting brain information transfer remains unanswered. However, MIPAC has the potential to add correlative evidence to this question by allowing a more complete description of PAC dynamics. Indeed, applying MIPAC to more complex data and relating it to behavior (reaction times, performance, *etc.*) could add compelling correlative evidence for the neurophysiological role of PAC. Appropriate modeling should also be carried out to disentangle the relationship of PAC to the measures involved in its computation (e.g., other spectral characteristics of the brain processes).

Since the MIPAC measure was developed within the framework of mutual information, we can interpret its moment-to-moment and trial-to-trial variability as indexing changes in the amount of information shared by the phase and amplitude signals. The fact that MIPAC can be used to estimate PAC with high temporal resolution is valuable. In an analysis of actual brain ECoG data, we were able to detect the appearance of PAC

time locked to face stimulus presentations and to demonstrate the use of MIPAC for event-related PAC estimation, giving an insight into the dynamics of the coupling process without collapsing the dimensionality of the data. However, as we will discuss shortly, any conclusions based on these results should be drawn carefully.

2.5.3 Limitations, caveats and future directions

One should be aware that MIPAC gives a measure of the degree of relationship of any sort between higher-frequency amplitude and lower-frequency phase time series. Thus, a MIPAC peak may not mark a high-frequency amplitude peak only, but might signal some other form of consistent relationship between the two time series.

An increasing concern in PAC reports is the inability of all PAC methods to deal with spurious CFC in non-stationary signals. Generally, non-stationary processes show spectral correlations between components of their Fourier expansions [29; 74; 78] that can be misinterpreted as CFC. For example, the Fourier decomposition of a non-stationary signal with sharp spike-like features may lead to the spurious conclusion that some high-frequency activity is coupled to slower rhythms. This has been called spurious CFC. Some authors [29] have suggested that statistical analysis using surrogate data may help in dealing with the ambiguity introduced by the spurious CFC. Alternatively, a causal analysis between the CFC spectral components may be able to overcome these limitations. The new MIPAC method proposed here, like traditional PAC metrics, do not solve the problem of spurious CFC. Thus, physiological interpretation of MIPAC results should be subject to stringent statistical analysis supported by evidence for its physiological plausibility and function.

By defining the MIPAC-image (Fig.2.15 A,E) we were able to visualize dynamics in the single-trial MIPAC estimates and to look for patterns – here, in particular, single-trial co-variations of event-related changes in high frequency power and low frequency phase that were otherwise invisible. Thus, the proposed method might be a more powerful tool

for exploratory analysis to gain insight into the evolution of the PAC process and its dynamic relationship to other electrophysiological measures.

The local information-theoretic measures used to create the MIPAC measures might also be applied to different types of cross-frequency coupling, since procedures we used made no assumptions as to the nature of the signals being analyzed. Thus, any type of CFC might potentially be estimated by a similar MI-based approach, perhaps simply by replacing the phase and amplitude signals with any pair of measures of interest applied to the same or to two different signals.

In addition to the potential problem of spurious coupling, we have identified three main limitations of the MIPAC approach. First, is that due to the scaling performed to compute the nearest neighbors in Section 2.3.6, MIPAC may be sensitive to outliers. For this, an alternative scaling may be used, or one may simply clean the data ahead of computing MIPAC. The fact that signals are band-pass filtered in a narrow band may also help to overcome the problem of the outliers. Second, like other PAC measurement methods, MIPAC does not provide information about the directionality of the phase-amplitude interaction (e.g., whether lower (phase) frequency oscillations appear to drive changes in the power of the high frequency signals and/or vice versa). A natural solution to this limitation within the framework of information theory might be to combine PAC measurement with transfer entropy estimation [40]. So far as we know, this has not yet been applied using local information measures.

The third limitation may be inaccuracy in MIPAC estimation arising from sub-optimal sampling of the phase and amplitude values across trials. For *Event-related MIPAC*, estimation of the number of neighbors within a phase-frequency full cycle window alleviates this problem. However, if the trial-to-trial variations in PAC time course vary across a wider interval than one cycle, MIPAC may be underestimated. Another possible way to compute event-related MIPAC might be first to compute MIPAC on the continuous signal (or on the concatenated set of signal epochs) using *Instantaneous MIPAC* and then

to extract MIPAC estimates in segments time-locked to events of interest. For this, some computational means would have to be implemented to handle the large dimensionality of the distance matrix needed to find the nearest neighbors. Fast nearest-neighbor search algorithms bypassing the need to compute the entire distance matrix exist and might be usable for this purpose (e.g., [79; 80]). However, this approach would not make use of the cyclostationarity assumption that the *Event-related MIPAC* approach capitalizes on, and so might work well only if local stationarity in the continuous (or concatenated) data can be assumed (see Section 2.3.6).

It has been widely demonstrated that in the hippocampus[4; 81–84], human medial temporal lobe [85] and in the visual system in both human and non-human primates, [5; 46; 86–89], low-frequency phase of local field activity can be coupled to high-frequency power. It is often assumed that PAC indexes transfer of information from low-frequency phase to high-frequency amplitude, i.e., low-frequency phase 'drives' high-frequency amplitude. However, most current methods for estimating PAC cannot distinguish the direction of this information flow (but see [90]). The possibility of bi-directional interaction is supported by a recent report on ECoG data from epileptic patients that the envelope of gamma oscillations drives alpha phase [91]. No doubt, estimation of the directionality of the coupling is important for proper interpretation of PAC. Given this evidence and the temporal description provided by MIPAC, a natural question would be how to explore the directionality of the phase-amplitude interactions underlying MIPAC.

Quite recently, a new method [90] has been developed to estimate cross-frequency coupling using non-linear auto-regressive models. This method has the potential to model causal cross-frequency interactions by estimating time delays in interactions between the different frequency components. Further, a recent report [92] has suggested a unified framework for data-driven causal inference that bridges both information-theoretic and auto-regressive methods. The report demonstrates the equivalence of Granger causality and transfer entropy under Gaussian assumptions on the variables. Information theoretic

measures, indeed, may provide a non-linear model-free approach suitable for PAC estimation. Thus, based on previous work [90] and on the findings provided by [92], one might propose a data-driven estimation method for CFC including causal analysis based on local transfer entropy [93] rooted in the same framework presented here.

2.6 Conclusions

Here we propose a novel method, MIPAC, for estimating the dynamics of phase-amplitude coupling (PAC) in (continuous) single-trial and (event-related) multi-trial data using a measure of local mutual information. Two mutual-information PAC (MIPAC) variants, *Instantaneous MIPAC* and *Event-related MIPAC*, were here validated by applying them to simulated data and then to actual cortical surface (ECoG) brain data. In both cases, results were consistent with results of published PAC estimators. The main advantage of the proposed MIPAC measures is their more highly time-resolved description of PAC process dynamics. This makes MIPAC suitable, in particular, for the analysis of either spontaneous or event-related dynamics in electrophysiological data.

2.7 Acknowledgements

Chapter 2, in full, is a reprint of the material as it appear in Ramon Martinez-Cancino, Joseph Heng, Arnaud Delorme, Ken Kreutz-Delgado, Roberto C. Sotero, and Scott Makeig. *Measuring transient phase-amplitude coupling using local mutual information*. NeuroImage 185 (2019): 361-378. The dissertation author was the primary investigator and author of this paper.

This work was supported by National Institutes of Health grant 5R01-NS047293-12 and by a gift from The Swartz Foundation (Old Field NY).

The authors would like to thank Joseph Lizier and Andre Fonseca for discussions on the mutual information estimator and Yasser Aleman-Gomez for helping with Fig. 2.10.

Many thanks as well to Johanna Wagner and Alejandro Ojeda for fruitful discussions and manuscript editing. Thanks also to the reviewers for their inputs and comments.

Chapter 3

What Can Transfer Entropy Tell Us About Phase-Amplitude Coupling in Electrophysiological Signals?

3.1 Abstract

Modulation of the amplitude of high-frequency cortical field activity locked to changes in the phase of a slower brain rhythm is known as phase-amplitude coupling (PAC). The study of this phenomenon has been gaining traction in neuroscience because of several reports on its appearance in normal and pathological brain processes in humans as well as across different mammalian species. This has led to the suggestion that PAC may be an intrinsic brain process that facilitates brain inter-area communication across different spatiotemporal scales. Several methods have been proposed to measure the PAC process, but few of these enable detailed study of its time course. It appears that no studies have reported details of PAC dynamics including its possible directional delay characteristic. Here, we study and characterize the use of a novel information theoretic measure that may address this limitation: local transfer entropy. We use both simulated and actual intracranial electroencephalographic data. In both cases, we observe initial indications that local transfer entropy can be used to detect the onset and offset of modulation process periods revealed by mutual information estimated phase-amplitude coupling (MIPAC). We

review our results in the context of current theories about PAC in brain electrical activity, and discuss technical issues that must be addressed to see local transfer entropy more widely applied to PAC analysis. The current work sets the foundations for further use of local transfer entropy for estimating PAC process dynamics, and extends and complements our previous work on using local mutual information to compute PAC (MIPAC).

3.2 Introduction

A hallmark feature of electrophysiological recordings of brain activity is the presence of rhythmic oscillations [9; 10]. Interaction between activity in different frequency bands has been associated with particular brain states and stimulus responses in humans in both healthy and pathological conditions, and more generally in mammalian brains [17]. Until recently, oscillatory dynamics in different frequency bands were, in effect, treated as being largely independent. It is now acknowledged that rhythms at different frequencies can interact temporally both within and between brain structures. More importantly, clearly-defined cross-coupling interactions between neural frequency bands appear across mammalian brain evolution, suggesting they may be supported by a universal evolutionary mechanism serving essential brain functions [10; 94]. These facts bring relevance to cross-frequency coupling (CFC) studies, and more importantly, to its most widespread and studied variant, phase-amplitude coupling (PAC).

In phase-amplitude coupling, the phase of a slower rhythm regulates changes in the amplitude of activity at higher frequencies, either within the same signal or between two recorded signals [17]. It is understood that high-frequency oscillations (HFO) emerge from and are topologically constrained within small brain functional areas. In contrast, larger generating areas and/or area-coupled networks are associated with slower rhythms [9]. A consensus view of brain PAC is that activity in disjoint frequency bands interacts by manifesting an architectural hierarchy: low-frequency oscillations manifest or express

synchrony within and coherence between large neuronal ensembles, while their phase regulates local changes in faster field activity within brief time/phase windows [10; 95]. This pattern of cortical rhythmic interdependence is believed to foster efficient, precisely timed information transmission flow across spatiotemporal scales [94]. Implicit here is also the idea that PAC reflects the causal influence of low-frequency phase on high-frequency amplitude.

PAC has been observed between various frequency bands, in multiple brain regions, under different task conditions and in multiple species (see Table 1 in [17]). In addition to the widespread prevalence of PAC in healthy brain process, links have been found between PAC and a variety of neurological pathologies (e.g., in epilepsy [18], Parkinson’s disease [20; 96], Alzheimer’s disease [21], mild cognitive impairment [25], schizophrenia [22; 23], and obsessive-compulsive disorder [24]).

Several methods have been proposed to estimate PAC. Although none has been established yet as the gold standard, three methods have been most often used by the scientific community: the Mean Vector Length Modulation Index (MVLmi) [28], the Kullback–Leibler Modulation Index (KLmi) [26], and the General Linear Model Modulation Index (GLMmi) [27]. These methods rely on the assumed covariation of the phase and amplitude time series to statistically estimate PAC presence and strength. One limitation of these approaches is their lack of time resolution. A new PAC estimation method based on mutual information, recently proposed by us in [17]), addresses this challenge. Another limitation of ‘*correlation*’-based PAC estimation is that it assumes that interactions between the two time series are instantaneous, thus missing the effect of any delayed interplay among frequency bands.

Delays among brain signals arise mainly due to intrinsic information propagation lags through brain circuitry. It has been demonstrated that the brain oscillatory processes are intimately related to these delays, which are believed to constitute an essential mechanism for inter- and intra-brain network synchronization [97]. The delay

coordinated activity has proven vital for normal brain function to such an extent that its disruption has been associated with pathologies including multiple sclerosis [98] and schizophrenia [99]. The ability to estimate interaction delays in brain signals may enable estimation of the directionality and causation in the interaction. Thus, the estimation of the dynamics of directed interaction in multi-scale PAC scenarios may help understand PAC’s functional significance.

Concepts from information theory (IT) have proven effective in addressing some current PAC measurement constraints (e.g., [17]). In addition to the advantage provided by the model-free assumptions in the estimation of IT quantities, two specific developments have made information theory especially suitable to addressing some current PAC limitations: (1) Introduction of transfer entropy [40] as a measure of predictive information transfer and interaction delay between time series; and (2) development by Lizier of theory and methods for estimating IT measures in a pointwise or *local* manner [100].

Here, we explore the use of local transfer entropy to study and characterize phase-amplitude coupling dynamics. We aim to provide an initial report of the use of local transfer entropy to study the temporal dynamics of PAC process interactions involving delays. In Section 3.3, we provide a general background on information theory and introduce the concepts and estimation techniques for transfer entropy as well as its local measure. We then address PAC estimation using transfer entropy (Section 3.4) and then present simulated and actual data results in Sections 3.5 and 3.6, respectively. In Section 3.7, we discuss the results and provide general observations on the use and interpretation of TE in the context of PAC.

3.3 Information theory and transfer entropy

Information theory background. A central quantity in Information Theory is the Shannon Entropy H . To introduce this key concept, let us assume two discrete random

variables X and Y with sets of values x and y , respectively, probability distributions $p(x)$, $p(y)$, conditional probabilities $p(x|y)$ and $p(y|x)$, and joint distribution $p(x, y)$. These quantities are related by $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$. The quantity $H(X)$, which is the average of the “log-surprise” $\log_2 1/p(x) = -\log_2 p(x)$ for an observation $X = x$ (see Equation (3.1)), represents the expected uncertainty associated with a measurement x of the random variable X :

$$H(X) = - \sum_x p(x) \log_2 p(x) \quad (3.1)$$

Shannon entropy can be extended to two random variables X and Y ; then, the joint entropy can be defined as in Equation (3.2):

$$H(X, Y) = - \sum_{x,y} p(x, y) \log_2 p(x, y) \quad (3.2)$$

It is also convenient to define the notion of conditional entropy as the average uncertainty about x that remains when the value of y is known (Equation (3.3)):

$$H(X | Y) = - \sum_{x,y} p(x, y) \log_2 p(x | y) \quad (3.3)$$

With these definitions in place, we can then formalize the mutual information (MI) between the random variables X and Y as a non-negative and symmetric measure defined in Equations (3.4) and (3.5). By convention, base 2 or e logarithms are used, giving units of bits or nats respectively.

$$I(X, Y) = \sum_{x,y} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \geq 0 \quad (3.4)$$

$$= H(X) - H(X | Y) \quad (3.5)$$

By assuming a third random variable Z —with sets of values z , probability distribution $p(z)$, conditional probabilities $p(x | z)$ and $p(y | z)$, and related to X and Y by $p(x, z) = p(x|z)p(z)$ and $p(y, z) = p(y|z)p(z)$ — one obtains the conditional mutual information (Equation (3.6)):

$$I(X, Y | Z) = H(X | Z) - H(X | Y, Z) \quad (3.6)$$

Transfer entropy. Now, let us assume the coupled physical system $\mathcal{X}$ and $\mathcal{Y}$, whose behaviour is described by the random process X and Y produces the time series $x_t = \{x_1, \dots, x_N\}$, $y_t = \{y_1, \dots, y_N\}$ at the discrete recording times $t \in \{1 \dots N\}$. With these definitions in place, Wiener’s principle of causality states that, if knowledge about the past of realizations of X and Y together allows one to predict the future of Y better than knowledge about the past of Y alone, then a causal influence can be assigned from the process X to Y [101]. In the information-theoretic framework, this principle can be reformulated as “*What information does the past of X provide about the future of Y that the past of Y did not already provide?*” [41]. The quantity capturing this principle, *transfer entropy*, was formalized by Schreiber [40] in terms of the conditional mutual information (Equation (3.7)):

$$TE(Y \rightarrow X) = I(X^+; \mathbf{Y}^- | \mathbf{X}^-) \quad (3.7)$$

Here, X^+ is a future random variable of the process X , whereas $\mathbf{X}^-$ and $\mathbf{Y}^-$ designate the reconstructed past state variables of the process X and Y , respectively. This quantity as stated in Equation (3.7) can be viewed as the *average* information contained in the past of the source signal $\mathbf{Y}^-$ about the next state X^+ of the destination that was not already contained in X past, $\mathbf{X}^-$. Current extensions of this formulation have been proposed by Wibral et al. [41] and Lizier et al. [102] by assuming: (1) An interaction delay u between the time series x_t and y_t ; (2) The times series x_t and y_t can be approximated

by a Markov process of order k and l respectively. With this assumptions, Equation (3.7) can be rewritten in a more general form [102] as in Equation (3.8):

$$TE_{Y \rightarrow X}^{(k,l)}(t, u) = I\left(X_t : Y_{t-u}^{(l)} | X_{t-1}^{(k)}\right) \quad (3.8)$$

Transfer entropy is a positive asymmetric quantity whose interpretation is still being debated. However, a consensus seems to be forming around the idea that the quantity provided by TE, far from being interpreted as “true causality”, may be a predictive information transfer [41; 103] or predictive causality. These are important concepts that are usually tangled in discussions of information transfers (see [103] for an in-depth discussion). The idea of causal effect may be assumed to mean the extent to which a source process directly drives the next state of a destination process [103]. This can be seen in the action of a falling row of dominoes. On the other side, predictive causality implies one’s ability to predict without committing to a belief in causal efficacy.

Local transfer entropy. Most of the information theory quantities currently in use (e.g., entropy, mutual information, transfer entropy) can be seen as an spatial or time average of more fundamental *local* or *pointwise* information bearing quantities. Local information theoretic measures characterize quantities from a specific, localized subset of measurements x and y of the random variables X and Y , rather than the associated average measure computed over all available data [39]. For example, local mutual information values $i(x, y)$ (Equation 3.9) may be averaged to compute overall MI $I(X, Y)$ (Equation (3.10)):

$$i(x, y) = \log_2 \frac{p(x | y)}{p(x)} \quad (3.9)$$

$$I(X, Y) = E_{X,Y} [i(x, y)] \quad (3.10)$$

Since the TE is just a conditional MI (see Equation (3.7)), local transfer entropy

can be defined as the pointwise conditional mutual information computed from an specific source state $y_{t-u}^{(l)}$ to a specific target event x_t conditioned by the event state history of the target $x_{t-1}^{(k)}$ [102] as in Equation (3.11):

$$te_{Y \rightarrow X}^{(k,l)}(t, u) = i \left(x_t : y_{t-u}^{(l)} | x_{t-1}^{(k)} \right) \quad (3.11)$$

While the TE is a strictly positive quantity, the local transfer entropy may be either positive or negative, indicating whether or not the source $y_{t-u}^{(l)}$ is providing informative information regarding the set $x_t, y_{t-u}^{(l)}, x_{t-1}^{(k)}$. When TE is negative, Lizier [104] pointed out that the source element is actually *misleading* about the state transition of the destination. Other instances where other causal information sources influence the destination, or in stochastic systems, may lead to negative TE values. Given this, we may not lose any clear insight into the interaction of our variables of interest by neglecting negative TE values, as we will do here.

One of the practical advantages of the local transfer entropy is that it provides information about the dynamics of the information transfer while at the same time allowing recovery of the interaction delay δ between the time series of the system analyzed as in Equation (3.12) [39]:

$$\delta = \underset{u}{argmax} \left(TE_{Y \rightarrow X}^{(k,l)}(t, u) \right) \quad (3.12)$$

These features will be exploited in this manuscript to analyze the dynamics of the PAC process.

Estimating transfer entropy. Estimation of transfer entropy is usually carried out through the use of mutual information and conditional mutual information estimation methods. The simplest and most widespread estimators for MI are extensions of algorithms to compute entropy based on straightforward plug-in evaluation of defining densities by its empirical estimates (called the *plug-in* estimator by [105]). Another popular

branch of entropy estimation methods use a similar principle but estimate the underlying densities by (1) kernel estimation methods (*KDE*) [106; 107] or (2) by taking advantage of the geometry of the space jointly formed by the support of the variables used in the computation, approximating the densities at the point x using the volume defined by a sphere encapsulating its K nearest neighbors (known as K-nearest neighbors, *K-NN* [59; 108; 109]). Despite its widespread popularity, this family of methods is known to have serious bias problems [65–69]. Recent advances in the development of MI nearest-neighbour estimators, specifically that proposed by Kraskov, Stogbauer, and Grassberger (KSG) [59] have provided an alternative to substantially improve the problem of the bias by providing a method that effectively bypasses the need to estimate densities. The KSG estimator builds on the nearest-neighbors-based Kozachenko and Leonenko entropy estimator (KL) [69], which Kraskov et al. modified to make the bias resulting from the nonuniformity of the densities in marginal spaces cancel each other. To do this, Kraskov et al. observed that, for any fixed K value, the distance to the K th neighbor in the joint space is larger than the distances to the neighbors in the marginal spaces, which lead to use different distances scales in the joint and marginal spaces when using the KL estimator for computing MI [59]. Consequently, Kraskov et al. recommended not to use a fixed value of K for the marginal entropy estimation and proposed a *K-NN* MI estimator in (Equation (3.13)):

$$I(X, Y) = \psi(K) - \langle \psi(n_x + 1) + \psi(n_y + 1) \rangle + \psi(N) \quad (3.13)$$

Here, N is the number of samples of X and Y , and ψ denotes the digamma function ($\psi(x) = \Gamma(x)^{-1} \frac{d\Gamma(x)}{dx}$). The terms $n_x(i)$ and $n_y(i)$ designate the number of samples falling into a strip of the marginal space of X and Y , respectively, defined by the distance to its K nearest neighbors. MI values $I(X, Y)$ are returned in nats. For a detailed derivation of the method, see [59] or [17].

The KSG estimator constitutes an effective non-parametric estimator of MI that is

data efficient (resolving structures down to the smallest possible scales), adapts resolution (binning scale changes according to the underlying data point density), and has minimal bias [59]—is indeed unbiased for independent variables. The neat formulation of the KSG estimator allowed Lizier [39] to extend it to compute local mutual information by unrolling the expectation ($\langle \dots \rangle$) in Equation (3.13), yielding Equation (3.14):

$$i(x, y) = \psi(K) - \psi(n_x + 1) - \psi(n_y + 1) + \psi(N) \quad (3.14)$$

An extension of this formulation has been proposed for the computation of local transfer entropy through the direct estimation of conditional mutual information in the form of Equation (3.15):

$$i(x, y|z) = \psi(K) - \psi(n_{xz}) + \psi(n_{yz}) - \psi(n_z) \quad (3.15)$$

In this manuscript, we used this formulation proposed by Lizier for the computation of TE [104] as implemented in the JIDT Toolbox [110].

Active information storage. From a corollary of Wiener’s causality principle in the IT context, it can be derived that, for the values of TE to be interpretable in the context of Wiener causality, it is necessary to ensure that a signal can optimally predict its own future behavior. It is then convenient to introduce an IT quantity to describe how well a signal can predict itself. This is the aim of the active information storage (AIS) introduced by Lizier [100], which defined a measure of how much of the information from the past of the process X is observed to be in use in determining its *next observation*. Formally, the AIS can be defined as the expected mutual information between realizations $x_n^{(k)}$ of the past state $X_n^{(k)}$ and the corresponding realizations x_{n+1} of

the process X (Equation (3.16))

$$AIS_x(k) = I(X_n^{(k)} : X_{n+1}) \quad (3.16)$$

Given a range of values for the state's length parameters k and l , we use the maximal values of AIS to estimate the optimal value of the state's lengths that ensures optimal signal self-prediction.

3.4 Approaching PAC estimation with information theory local measures

We now study and characterize the temporal dynamics of directed information transfer (from phase to amplitude) in the PAC process through local transfer entropy. Note that we assume only directed phase to amplitude interactions. This choice is discussed in Section 3.7.3.

Most methods for computing PAC follow a similar data processing pipeline. First, high and low-frequency band signals are extracted for the two frequency ranges in which the PAC coupling is to be assessed. These frequency bands are centered on a lower central frequency f_{phase} for the phase time series, and a higher central frequency f_{amp} for the amplitude time series. For this, either band-pass filtering or time-frequency decomposition can be used with similar results. Then, instantaneous phase and amplitude time series are obtained from the low- and high-frequency band signals, respectively, using the Hilbert transform or else directly from the complex signal when time-frequency decomposition has been used. These time series are then used to compute a PAC measure. After the PAC measure is computed, a statistical analysis is usually carried out by comparing the estimated PAC value with a distribution of surrogate PAC values calculated under a no-PAC assumption. Surrogate values are obtained by computing PAC from the original input signals after shuffling the two time-series many times as to destroy any PAC relationship [26].

Significance is then assessed by determining whether the estimated PAC measure for the actual data belongs or not to the distribution of surrogate PAC values. Here, we will follow this same preprocessing pipeline.

We have recently proposed and validated a method, MIPAC, to estimate time-resolved PAC using local mutual information [17]. Estimating PAC through MIPAC begins by computing two time series capturing instantaneous phase and amplitude in the two frequency bands of interest. Then, local mutual information is computed between these two signals following Equation (3.14). When computing Equation (3.14) in MIPAC, rather than the Euclidean norm used for the instantaneous amplitude, a circular norm [71] —to account for its periodic nature— is used for the phase component to compute and find the nearest neighbors in the marginal space defined by the support of the instantaneous phase. Finally, the local mutual information time series is low-pass filtered under f_{phase} .

Our analysis using local transfer entropy resembles that used for MIPAC [17]. Again, it begins by computing two time series capturing the instantaneous low-frequency phase and high-frequency band amplitude. To assess phase-to-amplitude information transfer and estimate the delay in the interaction between these frequency features, we first estimate the TE for a range of delays u . Given the periodic nature of the frequency components in question, a plausible range of delays may be defined by a time range smaller than the span of a full cycle of the lower (phase) frequency. Since the TE is maximal when the parameter u is the actual interaction delay δ [41], we choose to analyze the local transfer entropy time series corresponding to the u that maximizes the TE. Statistical significance is then computed at each latency as described previously.

Before TE computation, special consideration should be given to selecting the state history length for each signal. Although a few algorithms and methods have been proposed to estimate these lengths (e.g., [111; 112]), a consensus on choice of method is far from being reached [39]. In our work, we estimate the history lengths (also called ‘*embedding parameters*’) k and l by determining empirically the embedding values that

maximize the AIS in the instantaneous phase and amplitude. Since the KSG algorithm is bias corrected by construction, the values of k and l that maximize the TE should successfully capture the corresponding past states' relevant information. The estimated history values, along with an estimate of the nearest neighbors parameter K , are used then to estimate the delay u as described previously. In the current manuscript, data processing, simulations, computation, and analysis were performed using EEGLAB [57], functions from the PACTools plug-in to EEGLAB [113], and custom scripts written in MATLAB (The Mathworks, Inc.). The JIDT Toolbox [110] was used to compute local TE and AIS.

3.5 Simulation results

Simulating PAC. Simulated PAC signal coupling was generated by following [114]. Here, coupling in the signal was simulated between amplitude frequency $f_{Amp} = 70$ Hz and phase frequency $f_{phase} = 6$ Hz with a sampling frequency of $fs = 1000$ Hz. The lower frequency component, or modulator S_ϕ was built by band-pass filtering a Gaussian white noise signal around f_{phase} assuming a $\Delta f_{phase} = 1$ Hz bandwidth. The filter consisted in a Hamming-windowed (sinc) FIR notch filter implemented in the EEGLAB function *pop_eegfiltnew.m* [57]. The modulator signal was normalized to have unit standard deviation $\sigma_{phase} = 1$ before computing the cosine of its instantaneous phase obtained by using the Hilbert transform. The resultant modulator signal S_ϕ was then used to generate the high frequency component, or carrier S_A . For this, a sinusoid with frequency $f_{Amp} = 70$ Hz was modulated by using a sigmoid fed by S_ϕ as in Equation (3.17):

$$S_A(t) = \frac{1}{1 + \exp(-\lambda S_\phi(t))} \quad (3.17)$$

Here, the parameter $\lambda = 6$. We then introduced a delay between S_A and S_ϕ by shifting S_A τ ms forward respect to S_ϕ . In the remainder of the text, $\tau = 30$ ms if not

otherwise specified. Finally, to obtain the simulated PAC signal ($N = 5000$ samples), we added both the delayed S_A , S_ϕ and a Gaussian white noise with power of 0.5 dBW, as implemented in MATLAB function *wgn*, to yield a resultant signal with $SNR = 23.7$.

Simulation results. A continuous PAC simulated signal was generated as indicated at the beginning of this section. Figure 3.1 shows the time-averaged mutual information phase amplitude coupling (MIPAC) computed between phase frequencies from 3 to 10 Hz (1-Hz steps) and amplitude frequencies from 40 to 120 Hz (in 10-Hz steps) in the simulated signal. The figure shows that the coupling has been effectively introduced between amplitude frequency $f_{Amp} = 70$ Hz and phase frequency $f_{phase} = 6$.

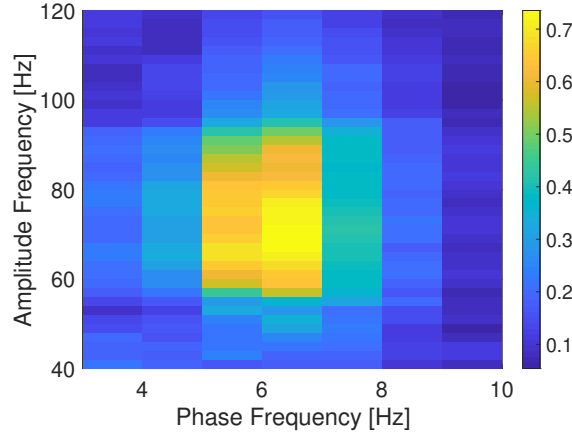


Figure 3.1. *Comodulogram for the simulated PAC signal.* Comodulogram depicting the time-averaged mutual information based phase amplitude coupling (MIPAC) computed between phase frequencies from 3 to 10 Hz (1-Hz steps) and amplitude frequencies from 40 to 120 Hz (10-Hz steps) for the simulated signal.

In Figure 3.2, we show the simulated PAC signal (top panel) and the time course of the MIPAC computed for $f_{phase} = 6$ and amplitude frequencies from 40 to 120 Hz (10-Hz steps) (lower panel). Black and red lines here depict the beginning and end of the high frequency oscillation (HFO) bursts. In this case, we see that from the perspective of the MIPAC, PAC seems to be present along the whole signal while missing some HFO around 1000 ms and 4000 ms. Roughly, the MIPAC seems to peak after HFO onset, but this is not strict, since MIPAC peaks appear following the HFO in some instances. In these

instances, MIPAC appears to be *reset* by the onset of the following HFO.

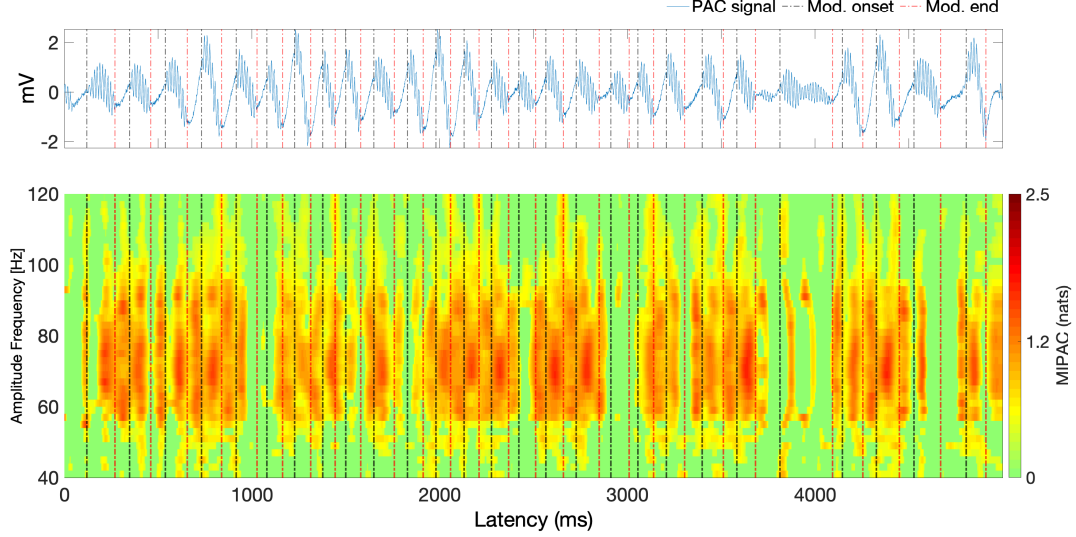


Figure 3.2. *Simulated PAC signal and MIPAC.* *Top panel* shows the simulated PAC signal with coupling between amplitude frequency $f_{Amp} = 70$ Hz and phase frequency $f_{phase} = 6$ Hz using a sampling frequency of $f_s = 1000$ Hz. Here, the phase component is delayed 30 ms with respect to the amplitude component. Vertical black and red dotted lines denote the onset and end of HFO bursts. *Lower panel* shows the MIPAC computed between $f_{phase} = 6$ Hz and amplitude frequencies from 40 to 120 Hz (10-Hz steps). Despite the delay between the frequency components, MIPAC seems able to detect the temporal evolution of PAC in the simulated signal. MIPAC appears to peak roughly by the end of the HFO events with some instances where the peaks persist after the culmination of the HFO. In these cases, a *reset* of the MIPAC appears to occur with the onset of the following HFO. All non-zero values are statistically significant ($p < 0.05$, uncorrected) as per a test performed using 100 surrogate values. Non-significant values were set to zero.

To characterize the self-prediction ability of the instantaneous phase and amplitude components, at $f_{phase} = 6$ Hz and $f_{Amp} = 70$ Hz respectively, for a given value of nearest neighbors $K = 50$, we compute the active information storage as a function of the history length parameter k . Unless otherwise specified, throughout the manuscript, significance testing was carried out by generating 100 surrogate signals whose duration and number of samples match those of the signal for which statistical significance is being assessed. The results of this computation are shown in Figure 3.3.

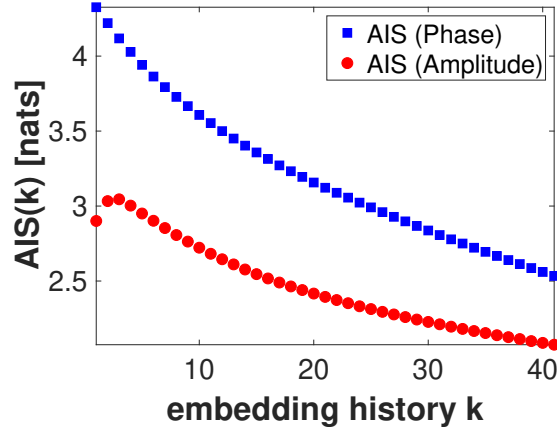


Figure 3.3. *Active information storage as a function of the embedding history k .* Active information storage (AIS) computed for the instantaneous phase at $f_{phase} = 6$ Hz (blue squares) and instantaneous amplitude $f_{Amp} = 70$ (red dots). Peaks in the AIS suggest that an embedding history of $k = 1$ for the phase and $l = 3$ for the amplitude are adequate to capture the relevant past history. All values shown are statistically significant ($p < 0.05$, uncorrected) as per a test performed using 100 surrogate values.

In Figure 3.3, all computed AIS values for both instantaneous phase and amplitude were statistically significant ($p < 0.05$, uncorrected) as per a test performed using 100 surrogate values. The maximum AIS values for instantaneous phase and amplitude appear to suggest that $k = 1$ for phase, and $k = 3$ for amplitude ensure an optimal self signal prediction. In the following, we will use these peaks on the AIS as values for the history length parameters for instantaneous phase ($k = 1$) and amplitude ($l = 3$). After testing with multiple values for nearest neighbors (not shown here), this result appears to be stable with respect to the parameter K . Next, for the given embedding parameters ($k = 1$ and $l = 3$) and delay $u = 30$ ms, we compute the TE in both directions between instantaneous phase and amplitude for a number of values of K (nearest neighbors). Significance testing was carried out by generating 100 surrogate values. Figure 3.4 shows the results of this computation. Here, only positive and significant ($p < 0.05$, uncorrected) TE values are shown. Note that, independent of the value of K used, transfer of information from phase to amplitude appears to be dominant, peaking at $K = 116$. In the remainder of this

analysis, the transfer of information from amplitude to phase will not be considered.

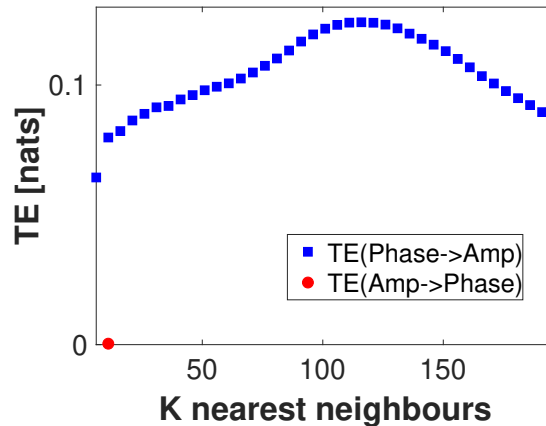


Figure 3.4. *Transfer entropy as a function of K -nearest neighbors.* Transfer entropy computed from the low frequency phase component at $f_{phase} = 6$ to the high frequency amplitude component at $f_{Amp} = 70$ (blue square), and in the opposite direction (red dots), as a function of the K -nearest neighbors values. Embedding history of $k = 1$ for the phase and $l = 3$ for the amplitude were used. Only significantly positive values are shown ($p < 0.05$, uncorrected). Information transfer from phase to amplitude appears to be predominant, independent of the value of K .

The lower panel of Figure 3.5 shows local TE computed from instantaneous phase at 6 Hz to amplitude in a band from 40 Hz to 120 Hz. The upper panel is similar to that in Figure 3.2, showing the simulated signal and the onset and offset of the HFO (red and black vertical dotted lines). For the TE computation, we used the parameters estimated in the previous analysis: $k = 1$, $l = 3$, $u = 30$ and $K = 116$. We can see that, despite similarities to the MIPAC results in Figure 3.2, the local TE seems to peak at HFO burst onsets and offsets. Similar to the behaviour of MIPAC in Figure 3.2, the local TE measure misses the HFO events occurring near 1000 and 4000 ms.

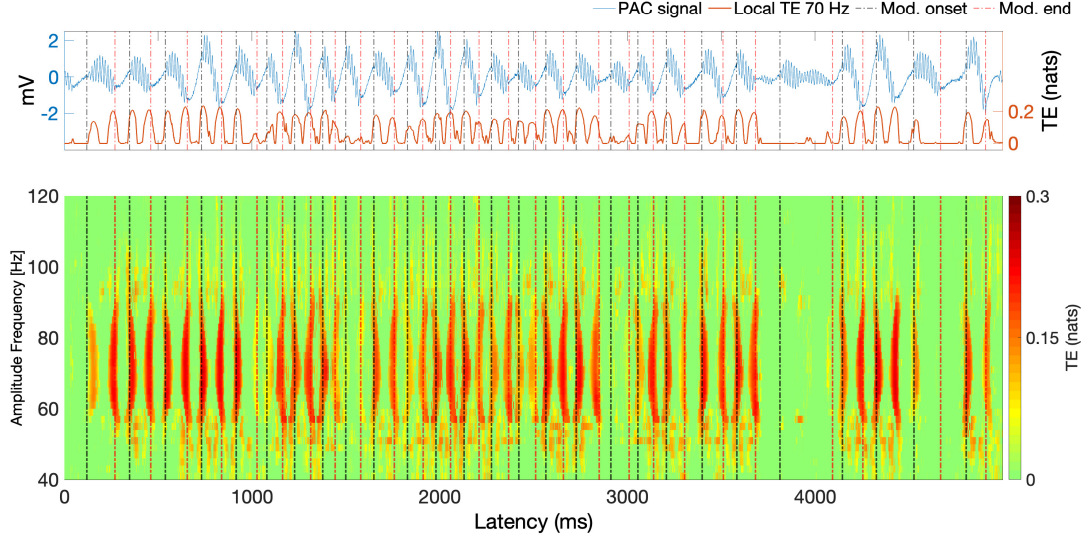


Figure 3.5. *Simulated PAC signal and local TE.* The *top panel* is similar to the top panel in Figure 3.2 and shows the original simulated signal in blue, the black and red vertical dotted lines marking HFO burst onsets and offsets. *Bottom panel* shows the local TE computed from the simulated signal’s instantaneous low frequency phase at $f_{phase} = 6$ to its instantaneous amplitude in the 40–120 Hz band. Significant TE values here appear at the beginning and end of HFO events. For reader’s convenience, we have added to the upper panel (see red solid trace) the time course of the computed local TE at 70 Hz. All non-zero values are statistically significant ($p < 0.05$, uncorrected) as per a test performed using 100 surrogate values. Before plotting, non-significant values were set to zero.

Finally, to test the capability of TE to recover different delays, we simulated a PAC signal with the same parameters as described at the beginning of the current section but with different delays u between the phase and amplitude components. These delays ranged from 0 to a delay corresponding to one full cycle of the phase frequency component at $f_{phase} = 6$ Hz (in this case, 166 time points). Delay estimation followed as described at the end of the Section 3.4, by using a range of delays ranging from 0 to 166 points and the parameters k , l , and K used previously. Figure 3.6 shows normalized TE values using a color code, as a function of the delay values used in the simulation (y -axis) and subsequent estimation (x -axis). The red dots denote the maximum TE achieved for each simulated signal given the estimated delay, and indicate the best estimated delay. Ideally, estimated and simulated delays would meet at the black diagonal dotted line. From our

results, the procedure followed appears to successfully recover the simulated delay, having a maximum deviation from the real simulated delay of only ten samples, equivalent to 10 ms (see Section 3.7.1 for further discussion).

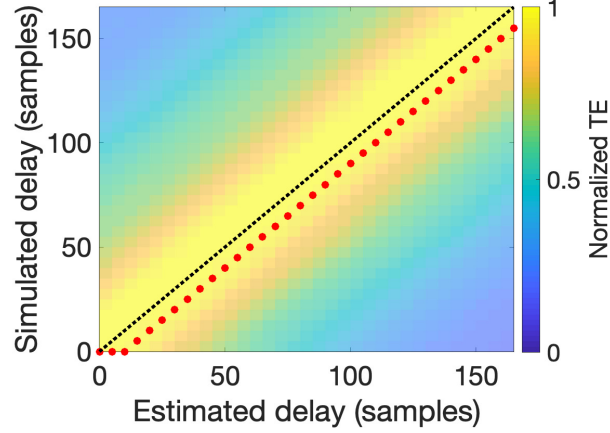


Figure 3.6. *Delay estimation.* Transfer entropy estimated using different delays ranging from 0 to the latency (166 time points) corresponding to one full cycle of the phase frequency component (here at $f_{phase} = 6$ Hz) for a set of simulated signals generated using the same delay values (y -axis). For each simulated signal, the maximum estimated TE value is indicated by a red dot; this yielded the best estimated delay. Estimated values appear to be consistently close to optimal estimation performance, indicated by the black diagonal dotted line.

3.6 Estimating PAC with transfer entropy from actual ECoG data

To evaluate our findings in real data, we applied the same methodology as in Section 3.5 to actual electrocorticographic (ECoG) data from a human subject.

Data collection. Electroencephalographic (EEG) recording from a single subject undergoing pre-surgical epilepsy evaluation at the North Shore University Hospital, Long Island Jewish Medical Center (NY) was performed using intracranial electrodes at a sampling rate of 1999 samples per second per channel. Data were referenced to common-average reference. Seizure detection algorithms were used; the data were also reviewed by an EEG technician and a physician. From a labeled data clip recorded during an epileptic

seizure, 5 s of data ($n = 9995$ samples) were extracted from a recording at an electrode (label: Tm2) located in the temporo-medial area (label: Tm2). Figure 3.7 shows the spectral characteristics of the data clip in an event-related spectral perturbation (ERSP) plot [73] obtained through a wavelet decomposition as implemented in EEGLAB function *newtimef.m*. We can see a rhythmic low frequency activation at 4–8 Hz simultaneously with a broadband (30–250 Hz) activation from this figure. We will refer to these frequency bands as low and high, respectively.

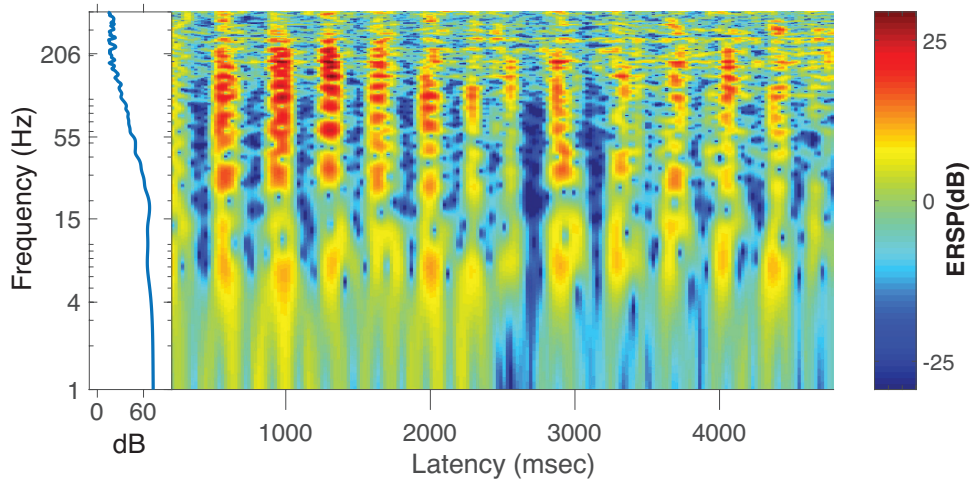


Figure 3.7. *Time-frequency features of the data clip.* Spectral characteristics of the 5 s data clip obtained through a wavelet decomposition implemented in EEGLAB function *newtimef.m*.

Computing MIPAC. Figure 3.8 shows MIPAC computed between the low (4–8 Hz) and high (30–250 Hz) frequency bands of the data clip shown in Figure 3.7. Instantaneous phase and amplitude were extracted from the low and high frequency ranges respectively as detailed in Section 3.4. Here, the time courses of the ECoG signal and of MIPAC are shown in red and blue, respectively. Significance computed using 100 surrogates ($p < 0.05$, uncorrected) appears in light gray. As we expected, increases in MIPAC correspond to HFO bursts appearing at a similar phase of the low frequency oscillation.

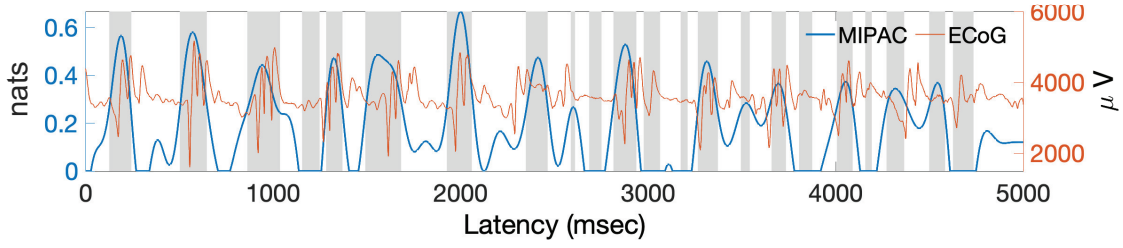


Figure 3.8. *Time course of ECoG data and MIPAC.* The red trace shows the 5 s time course for one ECoG data channel during an epileptic seizure. The blue trace shows MIPAC computed between low-frequency ECoG signal phase (at 4–8 Hz) and high frequency amplitude (30–250 Hz). Statistically significant MIPAC values ($p < 0.05$, uncorrected, 100 surrogates) are shown in light gray.

Estimating PAC using local TE. Using the same instantaneous phase and amplitude derived from the low- and high-frequency bands defined previously, analysis was carried out as in Figures 3.3 and 3.5. First, AIS was computed for embedding history lengths from 1 to 50 (not shown). Retaining the embedding history parameters corresponding to the highest AIS, their values were set to $k = 3$ (phase) and $l = 3$ (amplitude). Next, we computed transfer entropy in both directions, from phase to amplitude and from amplitude to phase, as a function of the K nearest neighbor parameter in the range $K = 1$ to 40 (Figure 3.9). The resulting information transfer was predominantly from phase to amplitude, independent of the value of K . However, some information transfer from amplitude to phase was also found.

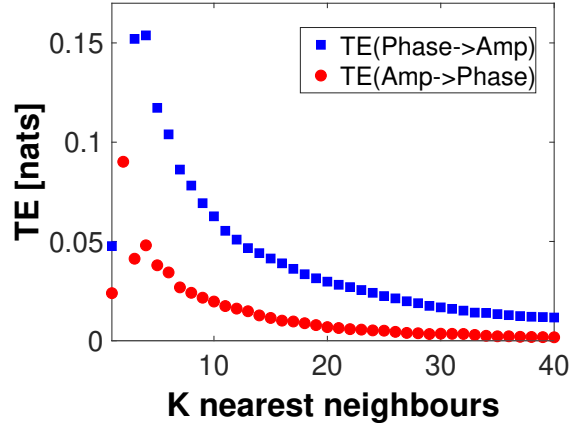


Figure 3.9. *Transfer entropy in ECoG data as a function of K -nearest neighbor values.* Transfer entropy computed on the 5 s data clip for a range of K -nearest neighbor values between instantaneous phase at 4–8 Hz and instantaneous amplitude in the range 30–250 Hz. Information transfer from phase to amplitude is designated with blue squares, while red circles designate transfer in the opposite direction. The process seems dominated by flow of information from phase to amplitude, although there is some flow of information in the opposite direction. Each value shown is statistically significant ($p < 0.05$, uncorrected) as per a test performed using 100 surrogates.

Next, we computed the TE from instantaneous phase to amplitude using the parameters k and l from the AIS analysis and $K = 100$, for a range of delay values ranging from 0 to 195 samples (corresponding to half a cycle of the central frequency, $f_{\text{phase}} = 6$ Hz, of the lower frequency band). The maximum TE in the delay range studied corresponded to a value of $u = 85$ samples (not shown). We selected this delay value along with the values of k , l and K previously used to compute the local transfer entropy from instantaneous phase to amplitude (Figure 3.10).

In Figure 3.10, the time course of the signal and the local TE are shown in red and blue, respectively. Significance testing ($p < 0.05$, uncorrected) for local TE, carried out using 100 TE surrogates, is shown in light gray. Local TE values were filtered below 12 Hz (roughly two times the central frequency of the lower frequency band) for better visualization of their main features. As can be seen here, local TE increases and peaks roughly at the beginning and end of HFO bursts. This result resembles that obtained for

the simulated signal in Figure 3.5. An apparent decline of TE values between 2000 ms and 3000 ms appears inconsistent with the MIPAC results shown in Figure 3.8.

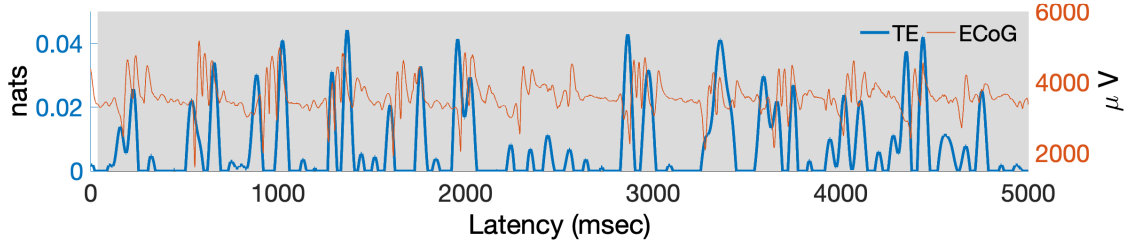


Figure 3.10. Time courses of the ECoG data and of local TE between low frequency phase and high frequency amplitude time series. The red trace shows the time course of the 5 s ECoG data clip recorded during an epileptic seizure. Computed local transfer entropy (from phase to amplitude) is shown in blue. The local TE time course seems to peak at onsets and offsets of HFO bursts in the ECoG signal. Statistically significant TE values ($p < 0.05$, uncorrected, 100 surrogates) are shown in light gray.

3.7 Discussion

3.7.1 Computing TE and MIPAC in simulated PAC data

In Section 3.5, we simulated a PAC signal (sampling rate, 1000 Hz) in which the instantaneous phase at $f_{phase} = 6$ Hz modulated the instantaneous amplitude at $f_{amp} = 70$ Hz. A delay of $u = 30$ samples (equivalent to 30 ms) was introduced to simulate a causal interaction from phase to amplitude. Using this signal, we computed mutual information-based phase-amplitude coupling (MIPAC) for all combinations of phase frequencies from 3 to 10 Hz (in 1-Hz steps) and amplitude frequencies from 40 to 120 Hz (in 10-Hz steps). A comodulogram conformed with the temporal average of MIPAC values (Figure 3.1) confirmed the existence of PAC at the selected frequencies. Comparing MIPAC time series to the time course of the simulated signal for $f_{phase} = 6$ Hz and $f_{amp} = 70$ Hz, we found that increases in MIPAC corresponded to bursts of simulated (70 Hz) HFO. Next, we computed bidirectional transfer entropy between phase ($f_{phase} = 6$ Hz) and amplitude ($f_{amp} = 70$ Hz). A critical step when computing transfer entropy is determining the signal's history length, or embedding parameters. Several methods have been proposed to estimate

these values, but none has produced a consensus as to the best estimation method. Here, we explored the space of parameters that maximize signal *self*-prediction, as measured by active information storage (AIS) [100], to obtain the embedding parameters we used for instantaneous phase (k) and amplitude (l) (see Figure 3.3). To compute the AIS, we assumed a value for the nearest neighbors of $K = 50$. However, the value of K seems not to be relevant, as AIS is very robust to the selection of this parameter [115]. Using the estimated parameters k and l , we computed transfer entropy as a function of the nearest-neighbor parameter K in both directions, from phase to amplitude and from amplitude to phase (Figure 3.4). We found that, in the simulated signal, the transfer of information from phase to amplitude was dominant, independent of the selected value of K . TE’s dependency on the parameter K seemed to be described by a concave function with a maximum at $K = 116$, which we used in further computations.

At this point, it is important to recall that TE estimation was carried out using the KSG estimator implemented in the JIDT toolbox [110], and, when using this estimator, TE is defined to be the average value of the local TE. In this case, we found that the K value corresponding to peak TE in Figure 3.4 was suitable for describing the process simulated, but using lower K values to estimate local TE led to increased variance in the time series. This fact brings to the table the issue that how to select the number of neighbors in the KSG algorithm is still under discussion.

Using parameters k , l , and K estimated in the previous analysis, we replicated the analysis in Figure 3.2 using local TE. Since in this case the flow of information *from phase to amplitude* was dominant (Figure 3.5), we only computed local TE in this direction. We found that, for a range of frequencies near $f_{amp} = 70$ Hz, phase at $f_{phase} = 6$ Hz is related to increased local TE near onsets and ends of HFO signal bursts. This result, as we can see, differs from the time course of MIPAC, which seems to most often reach its maximum local value during HFO bursts.

As it was demonstrated by Wibral et al. [41], TE is maximal when the delay

parameter u is equal to the true interaction delay δ [41]. This principle can be used to estimate the interaction delay between the process involved in the TE computation by exploring the space of the parameter u . In Figure 3.6, we simulated several signals using the same parameters as in Figure 3.2 but varying the delay between instantaneous phase and amplitude in a range from zero to 116 samples, corresponding to a full cycle of the phase frequency. For each of these signals, TE was computed for a range of u varying from zero to 116 samples. The TE maximum was indicated by red dots in Figure 3.6, while the true delay was indicated by the diagonal dashed black line. For each signal, we obtained a plateau of maximum TE values around the true simulated delay value, with the maximum deviation in the value of u being no bigger than 10 ms. It is obvious from Figure 3.6 that the maximum TE values always underestimate the real value and that an almost constant bias of 10 ms seems to be present. The cause of this bias is not clear and will be the subject of planned future study. However, we note that the 10-ms bias or deviation is on the order of the generation of action potentials [1].

3.7.2 Estimating PAC with local TE in actual ECoG data

To evaluate the use of local TE in estimating PAC in actual ECoG data, we used a brief recording from intracortical electrodes (ECoG) in a single patient undergoing pre-surgical monitoring for epilepsy (see Section 3.6). We used five seconds of data during a clinician-labeled seizure from a single electrode placed in the medial temporal brain region and referred to average reference across the ECoG recording array. The spectral characteristics of the signal in Figure 3.7) featured two main frequency ranges having significant activity, the low-frequency 4–8 Hz theta band and across a high-frequency broadband range (30–250 Hz) [116]. These two frequency ranges were used to first extract instantaneous (low-frequency) phase and (high-frequency) amplitude, and then to compute PAC using the MIPAC method (Figure 3.8). As in the simulated signal, statistically significant increases in MIPAC corresponded clearly to bursts of high-frequency activity in

the signal. This result was later compared to the local TE-derived PAC estimate.

To estimate local TE between these frequency ranges, we carried out an analysis similar to what we performed (Figure 3.3) to estimate the embedding parameters for phase (k) and amplitude (l), which suggested using $k = l = 3$. These values were used to study the relationship of bidirectional TE to the nearest neighbor parameter K (Figure 3.9). Contrasting with the simulated signal results (Figure 3.4), for the actual signal, we found statistically significant information transfer between phase and amplitude in both directions, though that from phase to amplitude dominated. This result was preserved over the range of K studied (up to $K = 100$). Next, we investigated the local TE for $K = 2$, where the TE between phase and amplitude in both directions was maximal, and confirmed that the variance of the local TE was considerably higher than for higher K values (not shown), and that TE features appeared to become independent of K for values above $K = 40$. Based on this, we decided to use the value $K = 40$ as well as $k = l = 3$ in our next computation. Then, we computed local TE from phase to amplitude using the same frequency components (Figure 3.10). The result replicated our finding for the simulated signal in which the local TE peaks occurred at high-frequency burst onsets and offsets. We speculate that these information transfer maxima at the beginnings and ends of high-frequency bursts reflect the causal role of the low-frequency oscillation controlling the appearance of the bursts, most likely indicating that firing of neurons in the local neighborhood is thereby constrained within a small time window (a few 10 s of ms) at times at which the relatively large theta frequency swings produced a biasing potential that encouraged neural spiking across neurons in the affected cortical domain. Thereby, spike impulses from the PAC-exhibiting area, upon reaching common target neurons in near synchrony, should be more effective in affecting the activity of the target neurons, possibly enhancing short and long-term plasticity in those neurons and shaping effective coupling strength of these network pathways [117–119].

3.7.3 The problem of directional causality in PAC

In our work, we have only assumed the flow of information directed from low-frequency phase to high-frequency broadband amplitude, disregarding the potential opposite interaction suggested recently by [91]. This choice is in line with an overwhelming amount of experimental work published in the last years indicating and providing models to support phase-to-amplitude coupling (e.g., [119–123]). For example, neurophysiological evidence for the modulation of high-frequency amplitude by the phase of slow neuronal oscillations has been observed in the interaction between slow neocortical oscillations and thalamocortical rhythmic burst firing (spindles) during slow-wave sleep [120]. Slow oscillations are composed of synchronized states alternating between hyperpolarization (down-state) and depolarization (up-state) and spread through the neocortex travelling to the thalamus, where they promote a pattern of increase and decrease in rhythmic burst firing. Thus, the spindles’ temporal evolution is continuously initiated, shaped, and terminated by slow-wave-promoted corticothalamic feedback (for reviews, see [119–121]), providing evidence for the modulation of high frequency amplitude through oscillatory phase.

Another example supporting phase-to-amplitude interaction comes from work of Schroeder and Lakatos, [122; 123] examining coupling between gamma band (30–100 Hz) amplitude and phase at delta (< 3 Hz) or theta (4–7 Hz) frequencies [9; 28]. Slow oscillations in the cortex can become entrained to external rhythms, thus aligning high excitability phases or up-states to occurring or expected external events so as to enhance their sensory processing. During the slow-oscillation high-excitability phase, gamma band (or high-frequency broadband) amplitudes may be enhanced such that a gamma burst occurs at the time when a task-relevant input is expected. Since gamma-band activity appears to be more metabolically-demanding than low-frequency oscillations [124; 125], the coupling between gamma and the slower phase is believed to ensure that information transfer resources are used efficiently and high-frequency activity (and concomitantly, spike

signaling using local neuronal resources) is selectively enhanced at critical time points. This suggests that slow oscillations act as gatekeepers for local high-frequency (and spiking) activity, thus suggesting a phase-to-amplitude causal relationship (for a review see [126]).

It is appropriate to recall that PAC has also been demonstrated in nonlinear oscillators [29] in which there is no specific modulation process or known mechanism for generating such an effect, the nonlinear dynamics of the system itself thus being the most probable cause of the PAC phenomenon. Therefore, PAC results (including those obtained through information theory-based methods) must be interpreted cautiously, especially if there is no physiologically plausible mechanism or model explaining the process. We acknowledge that a causal relationship from high-frequency amplitude to low-frequency phase might exist, but we argue that this should come from some common driving activity operating with different delays in the high- and low-frequency ranges in which PAC is assessed.

3.7.4 MI and TE, two faces of the PAC process?

In both our simulated and actual data analyses, MIPAC indicated a continuous modulation process with a local maximum within the time window of the modulated high-frequency bursts. On the other hand, TE peaks appeared at the beginnings and ends of these periods, though in some instances (see Figure 3.10, 2000–3000 ms), where MIPAC indicated a coupling, TE failed to display this behavior. These results pose the question of what information about PAC in cortex are provided by instantaneous MIPAC (a form of local mutual information) and by delay-estimating transfer-entropy based phase-amplitude coupling (TEPAC). Recall that local mutual information captures both linear and nonlinear statistical relationships between the two time series involved in its computation, while positive transfer entropy indicates a *weak* causality or predictive information transfer between them. These two principles are thus not in contradiction, nor are our PAC results. We hypothesize that the flow of information at the beginning

of the modulation process, as measured by local TE, initiates or facilitates the process that MIPAC measures. However, the mechanisms responsible for this facilitation remains unclear, and this idea may conflict with the implicit assumption that the PAC phenomenon is a continuous process carried out and determined by the influence of the timing of neural oscillations on a brain area’s cell ensemble. Another option is that TE may not represent a physical process in the PAC phenomena, and the results obtained are just a statistical characterization of the modulation process without further information on the actual biophysical interactions. In the current manuscript, we do not address these questions but rather here provide an initial report characterizing PAC computed using local transfer entropy. We do believe, despite these open questions, that the ability of TE to estimate interaction delays, adding to the overlapping information provided by MIPAC, may favor the use of local TE in addition to or instead of MIPAC for intensive PAC analysis. However, in our opinion, there are a few technical considerations that first must be addressed before recommending local transfer entropy as a method for studying PAC in electrophysiological signals.

3.7.5 General considerations for approaching PAC estimation using local transfer entropy

Parameters selection for estimating local TE

Perhaps the most controversial issue when computing transfer entropy is how to estimate its embedding parameters. As we did here, maximizing the AIS on each source to obtain the embedding parameters may potentially neglect multivariate effects between the past of both source and target signal. An alternative approach could be to optimize the AIS in the target signal and then find the source embedding that maximizes the TE from the source to the embedded target. In other words, one would find the source embedding that maximizes the conditional MI (CMI) between the source past and the next value in the target, conditioned on the target’s past. The CMI actually accounts

for both redundancies between both past states (e.g., due to a common driver/input to both signals), as well as potential synergistic effects. Such an approach has, for example, been discussed recently by Novelli [127]. It remains here to investigate how this strategy may improve TE estimation in our PAC context. However, as we have already discussed, there is not yet any consensus on the best way to approach the embedding parameter estimation. In addition, note that the exhaustive search method used in this manuscript may not be practical to use when studying PAC in lengthy continuous signals. In such cases, methods like that proposed by Ragwitz [112] that estimate the dimension and delay of the embedding while minimizing the prediction error for future samples of the time series, may be the better option. This method is implemented in two of the most advanced, specialized, and widely used software tools for computing transfer entropy and other IT measures: JIDT [110] and Trentool [128].

Currently, the only way to accurately estimate local IT measures including transfer entropy is via the KSG estimator which also requires setting another parameter, the number of neighboring points K in the joint space spanned by the signal supports used to define the marginal neighborhoods to compute Equation (3.14). Unfortunately, there is currently no efficient approach to estimating this parameter, though once it is set properly (to avoid undersampling of points in the marginal spaces), the computed measure (either local TE or MI) is quite stable to the selection of [115]. To summarize the discussion of these parameters, we would like to stress the key importance of the selection and setting of both the embedding parameters and the number of neighbors K for accurate computation of local TE.

Event-related data and TE

Here, we focused on characterizing the PAC process from the perspective of local TE applied to continuous signals. We did not consider the case of data segments time-locked to a set of similar stimulus events (i.e., event-related data epochs). We believe that these same

methods could also be applied to such event-related data, estimation in this case taking advantage of the data geometry inherent in the matrix of similarly latency-aligned data windows (dimension, number of trials by latencies), as proposed by Gomez-Herrero [70] and applied by us for MIPAC estimation [17].

The importance of the filtering strategy

The two frequency components entering the PAC analysis are entirely determined by the filter bandwidths, which in turn depend on the filtering strategy used to extract them [29]. Ultimately, the selection of the bandwidth appears to be a critical part of preprocessing for TE computation. We have identified three ways in which the filtering strategy may negatively impact the PAC analysis outcome and, in particular, PAC measured by transfer entropy.

The first problem area is in selecting the bandwidth of the phase signal passband in order to satisfy the Hilbert transform narrowband requirement (ideally an almost sinusoidal activity). The choice of bandwidth for the phase component is constrained by the condition of the signal having a meaningful phase and is, therefore, often correctly chosen to be narrow [29]. However, the validity of the narrowband assumption for extracting high-frequency amplitude in most published PAC studies is questionable, as variability in high-frequency activity in brain signals very often does not have a narrowband characteristic [114; 116]. Thus, using the Hilbert transform may yield non-meaningful amplitude estimations and thus potentially flawed PAC analysis [114; 129; 130]. On the other hand, shrinking the bandwidth to satisfy the Hilbert narrowband requirement has been shown to lead to misidentification by PAC of phase-frequency coupling—another form of cross-frequency coupling [131]. It has also been reported that narrowband filtering leads to an under-estimation of the information transfer delay [132]. In particular, Wollstadt reported a constant underestimation of the delay using narrowband filtered signals, a delay that increases the more narrow the band becomes (see Figure 7 in [133]). Thus, narrowband

filtering may also be a potential source of error when reconstructing delays using TE. Although further investigation is needed, we hypothesize that narrowband filtering could be responsible for the latency bias shown in Figure 3.6. While much of the PAC literature uses the Hilbert transform to estimate both the phase and amplitude signals, other Fourier-based time-frequency methods might equally be applied to extract the amplitude signal of wideband high-frequency processes.

The second problem area is in selecting the bandwidth to capture the higher-frequency spectral features indicating an ongoing PAC process. Here, the bandwidth used to extract the high-frequency component should be wide enough to accommodate the sidebands peaks indicating modulation by lower-frequency activity. This requirement is key for successful PAC estimation and has led to strong PAC guidelines suggesting that, if a narrowband higher frequency component’s bandwidth does not include the sideband peaks produced by amplitude modulation by the lower frequency, then the presence of CFC cannot be detected [29]. Thus, as per Aru and colleagues, the solution to these requirements is to look for a sweet spot where the frequency dynamics shows a meaningful activity against small bandwidth settings [29].

Finally, methodological problems associated with isolation of frequency band activity by filtering have been largely discussed in the context of Granger Causality (GC) [134–136]. These issues may also affect TE computation, which for jointly Gaussian variables has been demonstrated to be equivalent to GC [92]. In particular, Weber [132] has shown that, when analyzing connectivity in a network setting, filtering and downsampling signals ahead of computing TE may reduce the number of detected connections. We believe that most of these caveats may be overcome by approaching TE findings with a hypothesis-driven analysis supported by stringent statistical testing, but further investigation is necessary in this regard. A recently proposed method has recommended addressing the problem of filtering in TE computation by implementing the filtering only when generating surrogate data representing the null-hypothesis of no information transfer at the frequencies of

interest [137]. Studying how this approach will influence PAC analysis by local TE, where the signals entering the TE analysis are first bandpass filtered and Hilbert transformed (see Section Specific caveats in [137]), will be future work.

Here, despite the above-mentioned limitations, we have shown the potential of TE to address the study of delayed interactions in the PAC process. It should be noted that the same formulation used here for TE potentially allows studying PAC conditioned by other and/or more variables. This indeed may be a perfect approach to address the question of directionality between two processes when both may be driven by some third process. Although computing TE while naively conditioning on several other signals (e.g., instantaneous phase and amplitude values at other sources or channels) seems currently numerically infeasible [102], it is convenient to know that this may be an option in the future, given the ever-retreating compute horizon. In this regard, it is worth mentioning that recent proposals for the estimation of conditional and multivariate TE may ease the computational burden presented by the combinatorial explosions in the number of source combinations evaluated by implementing a non-uniform embedding strategy [127; 138–140]. These strategies may represent a possible solution for the current problem in the TE-resolved PAC when conditioning by several instantaneous phase and amplitude values at other sources or channels.

3.8 Conclusions

Here, we used local transfer entropy (TE) to estimate and characterize phase-amplitude coupling in cortical local field activity. We used first simulated and then actual ECoG seizure data and, in both cases, found local TE peaks at onsets and offsets of PAC modulation periods estimated using MIPAC, our previously reported method for estimating time-resolved PAC. Although further investigation is needed, we hypothesize that information transfer indicated by TE may signal, or even facilitate, the coupling

process. This mechanism should be the focus of further studies. We discussed some limitations we consider important to address before recommending that studies of PAC in electrophysiological signals rely on the use of local transfer entropy. Despite these cautions, we see real potential in the use of TE for the study of PAC, and specifically in the study of its interaction delays, which to date are an issue not widely discussed in the PAC literature. We are aware that, in our attempt to characterize PAC by features highlighted by local TE, we are leaving a number of open questions. We hope this initial investigation will help catalyze interest in the application of local TE to the brain PAC phenomenon, hopefully shedding light on the physiological role of PAC processes in the human brain.

3.9 Acknowledgements

Chapter 3 is a reprint of the material as it appear in Ramon Martinez-Cancino, Arnaud Delorme, Johanna Wagner, Kenneth Kreutz-Delgado, Roberto C. Sotero, and Scott Makeig. *What Can Local Transfer Entropy Tell Us about Phase-Amplitude Coupling in Electrophysiological Signals?*. Entropy 2020, 22(11), p.1262. The dissertation author was the primary investigator and author of this paper.

This work was funded by NIH grants R01NS047293 and R01MH123231.

The authors would like to thank Joseph Lizier for fruitful discussions and advice, and Ashesh Mehta for supplying the data used in Section 3.6. We are also thankful to The Swartz Foundation (Old Field, NY, USA) for its contribution to making this research possible. Finally, we truly thank the anonymous reviewers for providing key comments that helped to substantially improve the discussion of our results.

Chapter 4

Computing Phase Amplitude Coupling in EEGLAB: PACTools

4.1 Abstract

Phase-Amplitude Coupling (PAC) in electrophysiological signals refers to the transient interplay of activities in different frequency ranges, wherein phase in a low-frequency band and amplitude in a high-frequency band are in some way dependent. PAC phenomena have received increasing interest in neuroscience given the growing evidence of their apparent role in both normal and pathological brain processes. This interest has resulted in publication of a wave of methods for PAC estimation, each with its own advantages and drawbacks relative to others. Motivated by the widespread study of this phenomenon, most academic open source software environments for analyzing electrophysiological signals (e.g., Fieldtrip, Brainstorm, MNE) have implemented at least one method of PAC estimation. Here we describe an EEGLAB plug-in release, *PACTools* for MATLAB (The Mathworks, Inc.), that computes PAC in either continuous or event-related data using any of five methods for PAC estimation: Mean Vector Length Modulation Index, Kullback-Leibler Modulation Index, Phase-Locking Value, General Linear Model Modulation Index and Mutual Information PAC (MIPAC). *PACTools* uses parallelized code for efficient performance and offers built-in direct access to online high-performance computing resources made freely available for nonprofit research by the

Neurosciences Gateway (*nsgportal.org*). *PACTools* features intuitive graphic user interfaces and equivalent command line calls that make its use straightforward and seamless in the EEGLAB environment. We discuss toolbox implementation, architecture, dependencies, and the implemented methods of PAC computation and visualization.

4.2 Introduction

Phase-Amplitude coupling in electrophysiological signals. It is widely accepted and demonstrated that activity in different frequency bands in brain electrophysiological data need not be independent. Indeed, statistical relations between activities in different frequency bands, measured as several types of Cross-Frequency Coupling (CFC) in brain electrical activity has often been reported (reviewed in [53]). Although CFC may refer to any potential interaction between frequencies, phases, amplitudes (or other band-limited measures of one or more complex time series) [43], most experimental and theoretical work has focused on phase-amplitude coupling (PAC). In PAC, amplitude in a higher frequency band is modulated by, or linked to the phase in a lower-frequency band, either as measured within the same recorded signal or between two concurrent signals [17].

It has been shown that PAC is a common phenomenon, occurring in recordings from multiple species, in both healthy and pathological human subjects, in or between different frequency bands, brain regions, and spatial scales (for review see [17]). The functional role or roles of PAC in brain function are not yet well understood. A working hypothesis is that since high-frequency field activity is often more topologically localized than low-frequency activity [9], the coupling between higher and lower frequency bands in brain signals may function to synchronize, regulate, and implement multi-scale communication within and between brain areas and spatial scales [10; 15].

Software for computing PAC. The presumed importance and diversity of PAC phenomena have spurred development of several methods for its estimation (e.g., [17; 26–

28; 42; 52; 53]). No one method has yet emerged as ideal, each method having its own advantages and drawbacks. However, four methods have been reported most commonly: Mean Vector Length Modulation Index [28], Kullback-Leibler Modulation Index [26], Phase-Locking Value [42], General Linear Model Modulation Index [27]. Some or all of these methods are available in widely used open source software environments for analysis of electrophysiological brain data including Fieldtrip [30], Brainstorm [31], MNE [32]). A main limitation of these methods is that they obscure the temporal dynamics of cross-frequency coupling, although some methods have addressed this issue (e.g, [17; 52; 53]). Computing time may be another challenge limiting PAC analysis, as data dimensionality in neuroscience research, and therefore the number of potential measures to compute, is dramatically increasing [141].

MIPAC, EEGLAB and the Neuroscience Gateway. We recently developed a new PAC estimation method, using a measure of local mutual information, MIPAC [17], that can be used to estimate a time-resolved PAC measure in either continuous or epoched data. Since its introduction in 2004, EEGLAB [57] (scn.ucsd.edu/eeglab) has become the software environment most widely used for analyzing EEG by the cognitive neuroscience community worldwide [33; 34]. Most recently, the *nsgportal* (github.com/scn/nsgportal) plug-in to EEGLAB has brought free and convenient access to high-performance computing (HPC) resources at the Neuroscience Gateway from EEGLAB sessions running on any laboratory workstation or personal computer running MATLAB. These two developments have allowed us to build and provide a curated toolbox for computing PAC that also provides tools to implement HPC computation: the *PACTools* EEGLAB plug-in (<https://github.com/scn/PACTools>)

PACTools. Here, we describe an EEGLAB plug-in to compute PAC in continuous and epoched data, in either ICA source-decomposed or raw scalp-channel signals: *PACTools*. Currently *PACTools* has implemented five methods to compute PAC: Mean Vector Length Modulation Index [28], Kullback-Leibler Modulation Index [26], Phase-Locking Value

[42], General Linear Model Modulation Index [27] and the Mutual Information PAC [17]. *PACTools* supports parallel computation, when available, via the MATLAB Parallel Computing Toolbox and implements direct access to free HPC resources accessible through the Neuroscience Gateway (*nsgportal.org*). *PACTools* is seamlessly integrated into the EEGLAB platform and data structures and offers an intuitive graphical user interface (GUI) as well as command line functions for scripted use. Visualization tools, accessible from both GUI and command line, are useful for data exploration and publication.

Structure of the manuscript. This manuscript is organized as follows: First (Section 4.3), we introduce the *PACTools* Toolbox and discuss its implementation, architecture, dependencies, and the data structure used to store PAC results. Details of the computation and visualization of PAC results are also presented. In Section 4.4, we illustrate the use of *PACTools* through examples involving first simulated and then actual EEG data. Finally, in Section 4.5 we discuss the toolbox status, give some caveats concerning its current implementation, and comment on its future extension.

4.3 Introducing the EEGLAB plug-in *PACTools*

4.3.1 Implementation, architecture, and dependencies

The *PACTools* plug-in runs on MATLAB (The Mathworks, Inc., Natick, Massachusetts, United States). As a plug-in to EEGLAB, *PACTools* requires the EEGLAB toolbox (version 2019.0, or newer) to be in the user’s MATLAB path following installation as per the standard EEGLAB plug-in installation instructions (see scn.ucsd.edu/wiki/EEGLAB_extensions). In *PACTools*, PAC computation can also be redirected from the user’s internet-connected local computer to the HPC resources hosted at NSG. This feature is supported and managed by the *nsgportal* EEGLAB plug-in, which interfaces EEGLAB with NSG directly from within EEGLAB. To use this feature, the user should have *nsgportal* installed as an EEGLAB plug-in, and NSG user authentication for a valid NSG

account (see www.nsgportal.org).

Functions in *PACTools* follow EEGLAB conventions. There are two main sets of functions in *PACTools*, designated with prefixes *pop_* and *eeg_*. When called from the command line with some compulsory parameters missing, *pop_* functions launch a graphic user interface (pop-window) to allow the user to provide required parameters; else, fully-specified command line calls run directly. These functions usually call lower level *eeg_* functions that may also be called separately from the MATLAB command line window or MATLAB scripts and functions. *PACTools* includes two main *pop_* functions, *pop_pac*, and *pop_plotpac*, intended for computation and visualization of PAC respectively. These functions are designed to be the two main points of interaction with the user.

Function *pop_pac* manages PAC computation by gathering parameters and calling *eeg_pac*; lower level functions managing the computation using each of the methods implemented (see next section). This function also handles the redirection of PAC computation to the local computer or to HPC resources via NSG. Results of PAC computation managed by *pop_pac* are stored in the basic dataset structure/object in EEGLAB (*EEG*), from where they can be fed into *pop_plotpac* for results visualization. Function *pop_plotpac* leverages a set of *eeg_* functions designated to set parameters for visualization and to plot PAC results by means of four fundamental *eeg_* functions: *eeg_plotcomod*, *eeg_plotcomodt*, *eeg_plottfpac*, and *eeg_plottrialpac*. The plots generated by these functions are described in Section 4.3.5.

Most current methods for estimating PAC, given a signal (or two signals) of interest, follows a standard four-step procedure : (1) The frequency ranges of interest are defined - usually a low-frequency band (typically between 3 and 15 Hz) and a high-frequency band (typically > 30 Hz). (2) Instantaneous phase and amplitude time series are obtained from the low- and high-frequency band signals, respectively. These values can be computed either by applying the Hilbert transform to the filtered signals in the two separated bands, or via Fourier or wavelet-based time-frequency decomposition of the whole data. (3) The

obtained sets of concurrent instantaneous phase and amplitude values are then used to assess PAC. Here methods differ. In *PACTools*, instantaneous phase and amplitude values are extracted using wavelet time-frequency decomposition as implemented in EEGLAB functions *newtimef.m* and *timefreq.m*. (4) If requested, statistical significance of the returned PAC values is assessed nonparametrically. Typically, the obtained instantaneous phase and/or amplitude time series are divided into data blocks which are shuffled many times to create surrogate data. Statistical significance for the PAC results is then estimated by determining whether the computed PAC measure of interest for the actual data belongs (or not) to the distribution of surrogate data PAC measures [26].

4.3.2 Methods implemented in *PACTools*

PACTools implements five methods of PAC estimation that fit into the four-step scheme described in the previous section: Mean Vector Length Modulation Index (MVL-MI) [28], Kullback-Leibler Modulation Index (KL-MI) [26], Phase-Locking Value (PLV) [42], General Linear Model Modulation Index (GLM-MI) [27], and Mutual Information PAC (MIPAC) [17] and its Instantaneous (instMIPAC) and event-related (erMIPAC) variants. The first four methods here were originally developed to assess PAC in continuous signals.

A modification of the GLM-MI method proposed by Voytek et al. [52], enabled the computation of PAC on event-related data. Briefly, given an event-related-derived epoched set of concurrent instantaneous phase and amplitude values, the modified approach applies the GLM-MI method across trials to each trial latency. As a result, a ‘trial average’ dynamic description of mean PAC changes through the trial latencies is obtained. *PACTools* also uses this framework to compute through-trial PAC dynamics when epoched data is provided.

The MIPAC method proposed by Martinez-Cancino et al. 2019 [17] addresses PAC dynamics in both continuous and epoched data. Thus, all the methods implemented in *PACTools* can currently operate on either continuous or epoched signals. The dimension

```

1 >> EEG.etc.eegpac(1)
2 struct with fields:
3     dataindx: [1 1]
4     datatype: 1
5     labels: {'Chan1 - Chan2'}
6     params: [1x1 struct]
7     cache: [1x1 struct]
8     glm: [1x1 struct]
9     instmipac: [1x1 struct]
10    klmi: [1x1 struct]
11    mvlmi: [1x1 struct]

```

Listing 4.1. The code sample shows how *PACTools* stores its results. The last four fields of the structure contain the computed measures. The field *dataindx* contains the indices of the IC or scalp channel signals used to extract instantaneous phase and amplitude respectively. Field *datatype* indicates the type of data used to compute PAC -either independent component (IC) or scalp channel data. ICs or scalp channel labels for the results are stored in field *labels*. Parameters common to all the methods are stored in field *params*.

of the computed PAC values depends on the format of the input data and on the method used (see Table 4.1).

Table 4.1. Dimensions of the PAC results generated by the methods implemented in *PACTools*.

EEG data format	PAC Method	PAC result dimensions
Continuous	MVL-MI, KL-MI, PLV, GLM-MI	Single value
	instMIPAC	(1, #latencies)
Epoched	MVL-MI, KL-MI, PLV, GLM-MI	(1, #latencies, #trials)
	erMIPAC	

4.3.3 *PACTools* output format

When PAC is computed through *pop_pac*, the results of the computation are stored in the EEGLAB main structure/object *EEG*, in field *EEG.etc.pac.eegpac*. The MATLAB session code sample below shows how PAC values computed using the GLM-MI (*glm* in the structure), instMIPAC (*instmipac*), KL-MI (*klmi*), and MVL-MI (*mvlmi*) methods for a pair of data channels (phase from ‘*Chan1*’, amplitude from ‘*Chan2*’) are stored.

The field *dataindx* contains the indices of the IC’s or channels in the EEG set

used to extract the instantaneous phase and amplitude respectively to compute PAC. The labels of these data streams are stored in field *labels*. Field *datatype* indicates the type of data used to compute PAC: 2 = independent component (IC) data; 1 = scalp channel data. The set of parameters common to all the methods computed, including the phase (*freqs_phase*) and amplitude frequencies (*freqs_amp*), are stored in field *params*. The field *cache* stores serialized parameters (as per [142]) used in the current PAC computation.

The last four fields of the structure contain computed PAC measure results. As many of these fields will be defined as PAC measures computed for the dataset using the same parameter set. Each of these fields stores a structure containing the dimensions of the output PAC values, the PAC output values themselves, and results specific to the method computed. The field *cache* allows quick comparisons to parameters requested for a new PAC computation, enabling the toolbox to avoid recomputing already computed PAC measures. PAC computations for new IC or scalp channel pairings using the same set of other parameters are appended to the existing *EEG.etc.eegpac* structure (e.g., as structures *EEG.etc.eegpac(1)*, *EEG.etc.eegpac(2)*, etc.), while calls with new parameters will overwrite the whole *eegpac* structure, after giving the user a warning and opt-out opportunity.

4.3.4 Computing PAC using: *pop_pac*

Function *pop_pac.m* provides the front-end interface for GUI-launched PAC computation in *PACTools* and can also initiate processing when called with sufficient parameters from the command line. Figure 4.1 shows the *pop_pac* GUI with a typical set of PAC parameters entered.

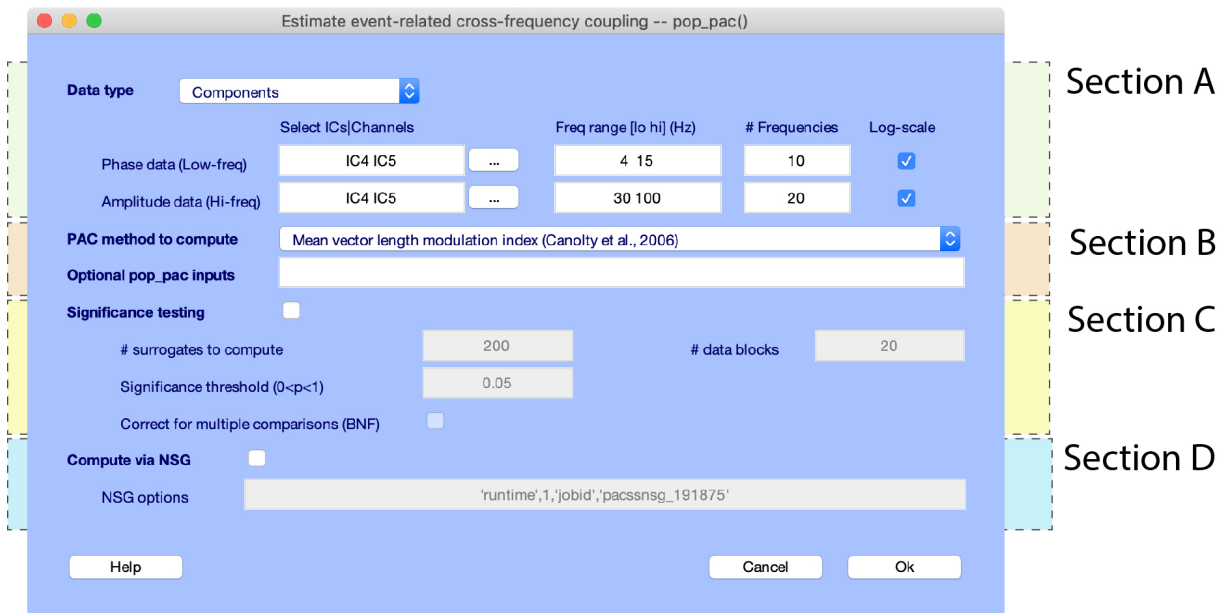


Figure 4.1. *The PACTools graphic user interface:* The GUI of *pop_pac*, the front-end interface for PAC computation in *PACTools*. The GUI has four main sections. *Section A*: Data and frequency settings. *Section B*: PAC method and *pop_pac* optional inputs. *Section C*: Statistical testing. *Section D*: Computing PAC via NSG.

This GUI can be invoked either from the EEGLAB interface by selecting menu item *Tools > PAC Tools > Estimate PAC*, or from the MATLAB command window by typing: *EEG = pop_pac(EEG)*. Here *EEG* is the currently loaded EEGLAB dataset. The *pop_pac* GUI is divided into four main sections.

Section A supports signal selection. **Data Type** specifies the type of signals used for PAC computation – either independent components (Components) or raw scalp channels (Channels). Below this are fields to select the (one or more) **Phase** and **Amplitude** signals to be used, their frequency passbands, and the number of passband frequencies to compute. By default, log frequency scaling is used to select these frequencies. The **Phase data** and **Amplitude data** fields must list the same number of components or channels. *PACTools* will thereby compute PAC for the listed combinations of phase and amplitude signals. For example, specifying IC4 and IC5 (as in Fig 4.1) as sources of both the phase and amplitude signals would direct *pop_pac* to compute PAC for $(IC4_{phase},$

$IC4_{amp}$), ($IC5_{phase}$, $IC5_{amp}$). To compute PAC between *different* ICs (or scalp channels), enter ‘IC4, IC5’ as **Phase data** and ‘IC5, IC4’ as **Amplitude data**.

Section B supports selecting the **PAC Method to compute** and allows input of **Optional *pop_pac* inputs**. To learn what input options are available, press *pop_pac.m* **Help**. Section C supports selecting available options for PAC **Significance testing** – the number of surrogate signals to compute (**# surrogates to compute**), the number of data blocks to use in shuffling the data to generate the surrogates (**# data blocks**) (see Section 4.3.1), the significance threshold to use (**Significance threshold (0<p<1)**), and whether to use Bonferroni correction for multiple comparisons (**Correct for multiple comparisons (BNF)**).

Section D supports performing the PAC computation on HPC resources of the U.S. XSEDE network via the Neurosciences Gateway (**Compute on NSG**). We will detail this capability in the next sections. Three buttons at the bottom of the GUI launch a help document window (**Help**), remove the *pop_pac* window without further action (**Cancel**), or begin PAC computation (**OK**). Starting *pop_pac* by pressing **OK** initiates a *pop_pac* command line call that is then also stored in the EEGLAB command history. EEGLAB function *eegh* allows users to retrieve the call code for future use in command line calls or in new EEGLAB scripts, functions, or plug-ins.

4.3.5 Visualizing PAC results

Visualizing results obtained from PAC computation in *PACTools* is implemented in function *pop_plotpac*, which can be called either from the command line (*pop_plotpac(EEG)*) or the EEGLAB menu (via menu item **Tools > PAC Tools > Estimate PAC**). A set of four standard plots are available in *pop_plotpac* to visualize the computed modulation indices for the implemented methods: (1) *Comodulogram* plots modulation indices as a function of amplitude and phase frequencies (calls *eeg_plotcomod*). (2) The *Time-comodulogram* plot adds the dimension of trial latency. [Implemented in *eeg_plotcomodt*].

(3) *Time-frequency PAC* plots modulation indices for a range of either phase or amplitude frequencies as a function of latency. This and the *Time-comodulogram* plots are only available for PAC dimensions (1, #latencies) and (1, #latencies, #trials) (see Table 4.1). [Implemented in *eeg_plotfpac*]. (4) *Trial-based PAC* plots modulation indices for all trials as a function of time. This visualization is only available for PAC computed using event-related MIPAC, which generates PAC results with dimensions (1, #latencies, #trials). [Implemented in *eeg_plottrialpac*].

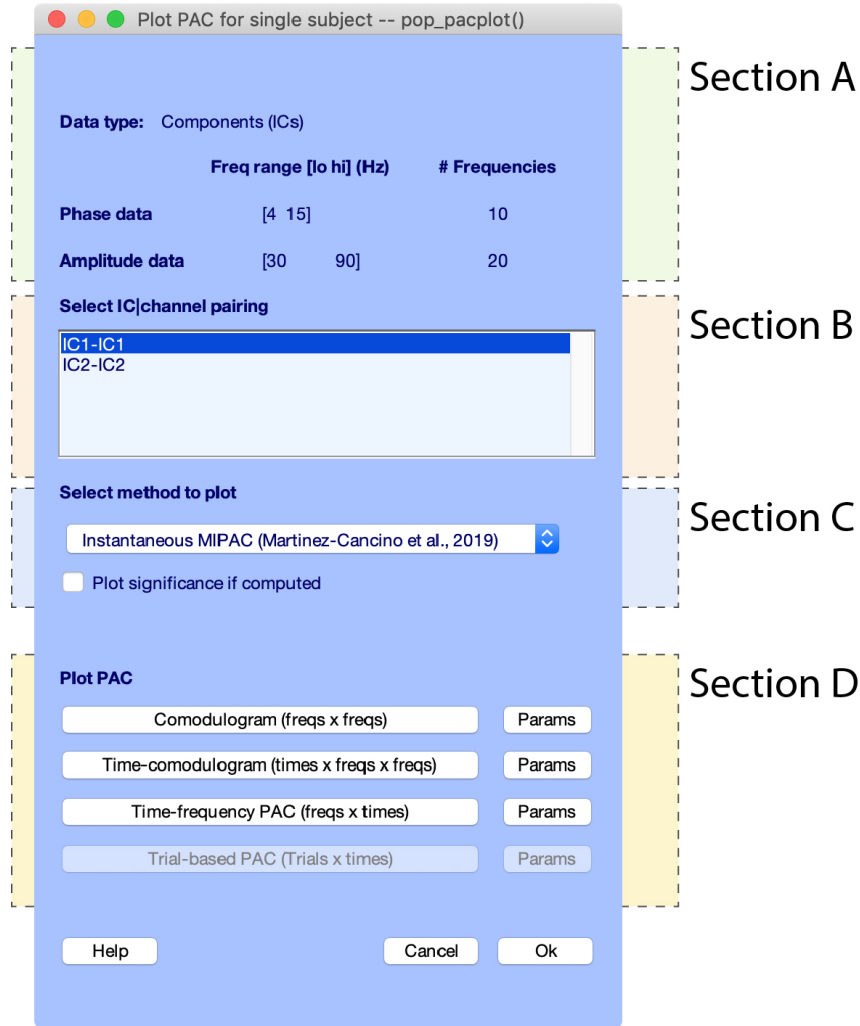


Figure 4.2. *pop_plotpac* GUI. The Figure shows the pop-up GUI for *pop_plotpac*, which presents an overview of the PAC parameters used (*Section A*), e.g., which IC or scalp channel signals were used in the computation (*Section B*), and the computed PAC method results to plot (*Section C*). *Section D* enables selection of plot type and, if needed, its detailed parameter settings. Plot types not available for plotting the currently computed PAC method results are greyed out.

The *pop_plotpac* GUI comprises four major sections (Fig. 4.2). Section A gives the user an overview of the parameters used in computing PAC (and then stored in the EEG dataset). Section B lists the component or channel pairs for which PAC has been computed. The first element of the pair was used to derive instantaneous phase, the second, instantaneous amplitude. The PAC methods for which results are available are shown in Section C. The user must select the set of method results to plot. A checkbox in this

Section enables plotting a significance mask on the PAC results (if this was computed). Section D allows the user to select plot type using the four buttons on the left side of the interface. Unavailable plot types are greyed out (see, in Fig. 4.2, **Trial-based PAC**). To the right, **Params** buttons enable the user to specify parameters including frequency and latency ranges to plot and other options (see Section 4.4.2). Each plotting option selected is stored in the EEG structure (field *EEG.etc.pacplotopt*) for possible future reuse and to insure reproducibility.

4.3.6 Code parallelization and high-performance computing

In *PACTools* we have implemented the capability to outsource the PAC computation to HPC resources on the U.S. XSEDE network via NSG. This feature may be enabled in a command line call to *pop_pac* using the parameter ‘*compflag*’ with argument ‘*nsg*’, or from the *pop_pac* GUI by checking the **Compute on NSG** checkbox (see Fig. 4.1). If the user chooses to perform the PAC computation through NSG, the function *pop_pac* will: (1) create a temporary NSG job folder, (2) copy the input data into this folder, and (3) generate and then save in this folder an NSG MATLAB script (*pacssnsg_job.m*) calling *pop_pac* with the user-specified parameters. Then *pop_pac* will submit this job folder to NSG using *nsgportal* command-line tools, identifying *pacssnsg_job.m* as the main script to execute. After submitting the job, *pop_pac* will exit, returning the NSG job ID assigned to the task. Later, the user can provide this job ID to *pop_nsg* to monitor job progress and download the results, once complete. See [34] for more details.

4.4 Proof of concept application: Computing and visualizing PAC on simulated data

4.4.1 Computing and visualizing PAC on simulated data

As a simple example of the application of *PACTools*, we computed PAC for a simulated EEG signal, a synthetic amplitude-modulated (AM) signal generated using

time-varying modulation. Here (Fig. 4.3C) signal amplitude at 60 Hz was modulated by phase at 8 Hz during two brief latency windows in a simulated EEG trial (sample rate, 500 Hz). PAC was assessed by erMIPAC for 10 (log-spaced) phase frequencies between 4-15 Hz and 20 (log-spaced) amplitude frequencies between 30-100 Hz.

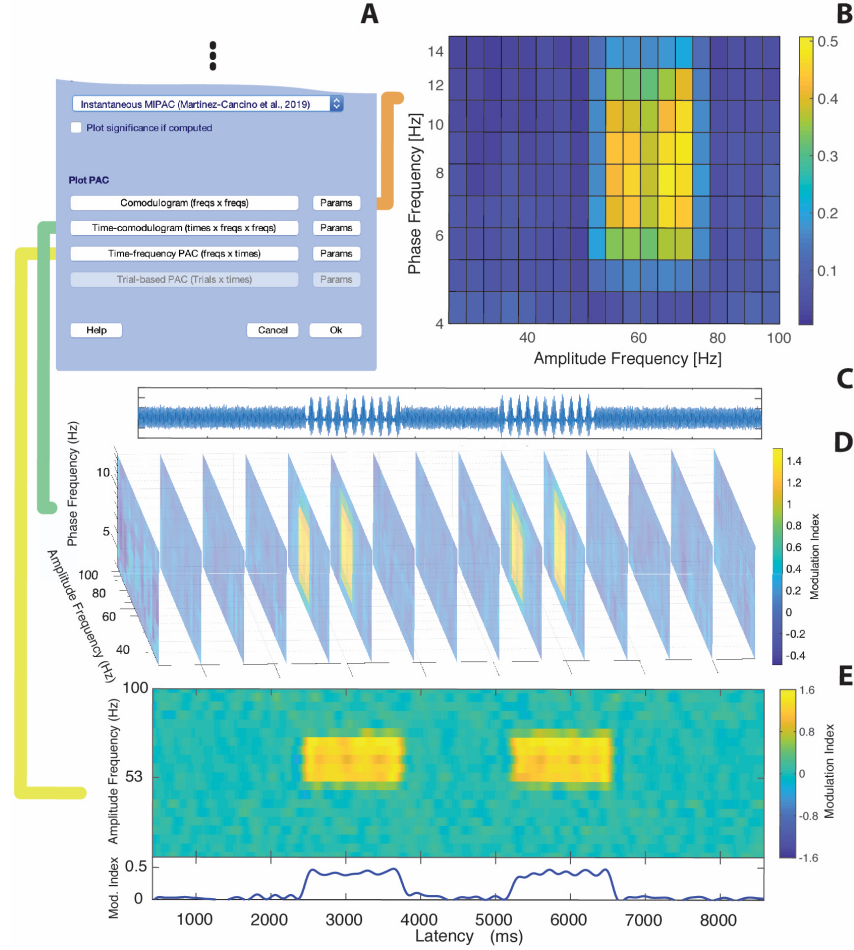


Figure 4.3. Visualizing PAC results using *pop_plotpac*: (A) Detail of the lower section of the *pop_plotpac* GUI, here as used to produce the indicated plots of the results computed in Section 4.4.1. (C) the simulated signal; (B) the comodulogram, (D) the latency-resolved comodulogram (**Time-comodulogram**) and (E) the latency-resolved PAC frequency plot (**Time-frequency PAC**).

Once the computation is completed, results are displayed from *pop_plotpac*. Figure 4.3A shows a detail of the lower section of *pop_plotpac* GUI. The upper section of the GUI (not shown here) indicates that the PAC computation has been effectively computed for the parameters and settings described in the previous paragraph. Panels B, D and E in Fig.

4.3 shows the (phase freq, amp freq) comodulogram, the latency-resolved comodulogram, and the (amp freq, latency) latency-resolved comodulogram for phase frequency 7 Hz and log-scaled amplitude frequencies 30-100 Hz. As we can see these visualizations capture different aspects of the modeled process and thus provide a convenient tool for exploring the results of PAC estimation.

4.4.2 Computing and visualizing PAC on actual EEG data

To assess the performance of *PACTools* on real data, we computed PAC in a single dataset from the study published by Wakeman and Henson [143] (available at *OpenNeuro.org*; accession number ds000117). The data used here are a set of 229 3-sec EEG data epochs (70 channels, sample rate 250 Hz) time-locked to presentations of images of famous faces, saved as an EEGLAB dataset [34]. For more information on the experiment and preprocessing steps applied to these data see [34]

Here, we computed PAC for a single independent component process (IC8) whose scalp map (Fig. 4.4B) is compatible with a predominant origin in left lateral occipital cortex, and which contributed strongly to the left inferior posterior peak P1 in the trial-average scalp-channel event-related potential (ERP) [144; 145]. We computed PAC using GLM-MI for 10 log-spaced phase frequencies (range, 4-15 Hz) and 20 log-space amplitude frequencies (range, 25-50 Hz). From the *pop_plotpac comodulogram* (not shown), we identified significant PAC with a maximum at (7-Hz phase, 35-Hz amplitude). Figure 4.4A shows PAC values (at 7-Hz phase frequency) across trial latencies and amplitude band frequencies.

Plotting modes useful for assessing the physiological validity of PAC results are available from the *pop_plotpac* GUI via the plot *Params* options (Fig. 4.2). For example, in Fig. 4.4 we show three Freq-time PAC plot modes: (1) Fig. 4.4B: The IC8 phase-sorted trial data corresponding to the component or channel used for obtaining the amplitude values for computing PAC. Here IC8 was used to extract both phase and amplitude values

at the frequency ranges defined at the beginning of this subsection. (2) Fig. 4.4C: The time course of instantaneous phase at 7 Hz for the trials shown in 4.4B. 3) The time course of instantaneous 35-Hz amplitude for the trials shown in 4.4D. Here we see that the peak in PAC values in 4.4A near 100 ms is produced by stimulus-evoked activity (contributing to the left ‘P1’ peak in the scalp ERP) with a 7-Hz characteristic and only slightly varying peak latency (or, phase relative to stimulus onset) (4.4C) and a concurrent consistent peak followed by a sharp decrease in 35-Hz amplitude.

This suggests that the PAC peak in 4.4A is produced in this localized IC process by two separable though concurrent stimulus-related changes in its cortical field signal: 1) production of a stimulus-evoked theta-band complex at only slightly varying latencies trial-to-trial, and 2) a stimulus-induced increase in ~ 35 -Hz activity followed by its abrupt blocking beginning at about 100 ms, although in single trials the latency of the 35-Hz peak and of its blocking are in fact here not clearly phase-locked to the evoked theta complex activity in the same trials.

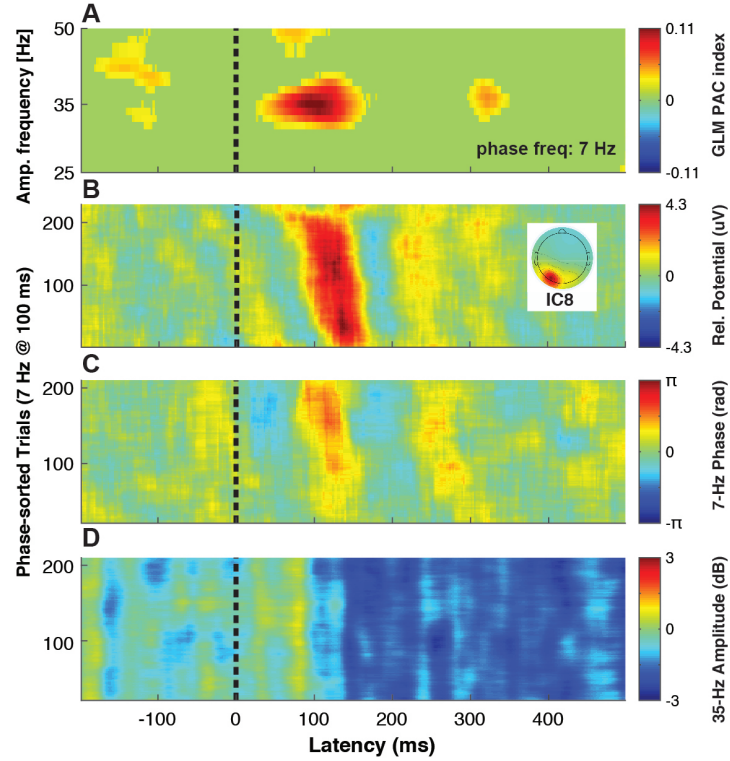


Figure 4.4. Assessing PAC on selected independent components of ICA-decomposed EEG data. PAC was computed by GLM-MI for IC8 (scalp map in inset) using 10 log-spaced phase signal frequencies in the range 4-15 Hz and 20 log-spaced amplitude signal frequencies in the range 25-50 Hz. Dashed lines at 0 ms indicate face picture presentation onset. These latency-resolved plots then show: (A) trial-latency evolution of PAC for phase frequency 7 Hz. Points with statistically non-significant PAC estimates computed out of 500 surrogates ($p < 0.05$, uncorrected) are shown in green; (B) the IC8 trial signals, sorted in order of their 7-Hz phase (computed in a 3-cycle window centered at 100 ms), with (vertical) 40-trial smoothing; (C) time course of instantaneous phase at 7 Hz for trials sorted and smoothed as in (B); (D) time course of instantaneous amplitude at 35 Hz for trials sorted as in (B).

4.5 Discussion and conclusions

The open-source EEGLAB plug-in *PACTools* is freely available to download at: github.com/scn/PACTools. Here we have shown detailed information on *PACTools* implementation and workflow, as well as examples of computing and then plotting the computed PAC indices on simulated and actual EEG data. The *PACTools* plug-in offers command-line and GUI capabilities for computing PAC from EEG or similar data represented as its independent components or as single scalp channels. Currently, *PACTools*

implements five PAC methods: Mean Vector Length [28], Kullback-Leibler [26], Phase-Locking Value [42], General Linear Model [27] and Mutual Information PAC [17]. The extensive documentation and help messages in *PACTools* insure that more methods that fit the four-step PAC-computation scheme described by the end of Section 4.3.1 can be easily added in future. Extending it to include PAC methods not relying on instantaneous phase and amplitude measures for coupling assessment might not be as straightforward.

Computing PAC on actual brain data can be computationally expensive. This is especially true when hundreds of surrogates have to be generated to estimate the statistical significance of PAC values, and/or using MIPAC methods involving estimation of local mutual information. With this in mind, *PACTools* is implemented using parallelized MATLAB code and includes EEGLAB *nsgportal* plug-in tools that make it simple to outsource more computationally intensive PAC computations to HPC resources hosted by the Neuroscience Gateway (*nsgportal.org*). *PACTools* thus provides accessible tools for computing PAC on electrophysiological data without the need to overstrain locally available compute resources.

In the PAC literature, visualization of the modulation indices uses a variety of plots often used in reports using the PAC methods that *PACTools* implements. In the literature to date, *comodulogram* plots showing the strength (or modulation index) of PAC coupling as a function of phase and amplitude frequencies are most common. *PACTools* extends these plots to a third dimension, trial latency, though the ability of *pop_plotpac* to visualize user-specified ‘slices’ of PAC results matrices. For event-related trial data, plot types that visualize relationships between instantaneous phase and amplitude are also available under plot *Params* (Fig. 4.4). These plots are useful for assessing PAC validity and its possible mechanisms.

We are aware the plotting approaches implemented to date may not fulfill all possibilities suggested by users’ curiosity and creativity. However, *PACTools* saves all method-related intermediate and final results for use in further assessments. For example,

as documented at github.com/sccn/PACTools, the GLM-MI PAC method uses linear regression to model the relationship between instantaneous phase and amplitude. In *PACTools*, PAC computed by GLM-MI stores the Beta values from each regression. By saving intermediate results for all the implemented methods, *PACTools* supports users wishing to develop customized visualizations and results exploration approaches.

Although *PACTools* command-line capabilities allow users to implement scripts and functions to process data from more than one subject, currently *PACTools* implements PAC computation for a single dataset at a time. Overcoming this limitation will be the goal of next extensions and releases. To date, *PACTools* core functions from its beta versions have been successfully used by the scientific community to investigate PAC in EEG recordings from healthy and pathological subjects (e.g., [17; 23; 146; 147]). We hope the availability of *PACTools* may contribute to extend its use, and to deeper the investigation of PAC phenomena in brain signals and their functions.

4.6 Acknowledgments

Chapter 4 is a reprint of the material as it may appear in Ramon Martinez-Cancino, Arnaud Delorme, Ken Kreutz-Delgado, and Scott Makeig. *Computing phase amplitude coupling in EEGLAB: PACTools*. Proceedings of the 20th International IEEE Conference on BioInformatics And BioEngineering, October 26-28, 2020. The dissertation author was the primary investigator and author of this paper.

This work was supported by National Institutes of Health grants R01-NS047293 (EEGLAB) and R01-EB023297 (Open EEGLAB Portal), and by a gift to UCSD from The Swartz Foundation (Old Field, NY).

We thank Joseph Heng for his contributions during early stages and The Swartz Foundation (Old Field, NY) for its contribution to making this research and tools possible.

Chapter 5

The Open EEGLAB Portal Interface: High-Performance Computing with EEGLAB

5.1 Abstract

EEGLAB signal processing environment is currently the leading open-source software for processing electroencephalographic (EEG) data. The Neuroscience Gateway (NSG, nsgportal.org) is a web and API-based portal allowing users to easily run a variety of neuroscience-related software on high-performance computing (HPC) resources in the U.S. XSEDE network. We have reported recently [148] on the Open EEGLAB Portal expansion of the free NSG services to allow the neuroscience community to build and run MATLAB pipelines using the EEGLAB tool environment. In this chapter we report on the release of an EEGLAB plug-in, *nsgportal*, that interfaces EEGLAB with NSG directly from within EEGLAB running on MATLAB on any personal lab computer. The plug-in features a flexible MATLAB graphical user interface (GUI) that allows users to easily submit, interact with, and manage NSG jobs, and to retrieve and examine their results. Command line *nsgportal* tools supporting these GUI functionalities allow EEGLAB users and plug-in tool developers to build largely automated functions and workflows that include optional NSG job submission and processing. Here we present details on *nsgportal* implementation

and documentation, provide user tutorials on example applications, and show sample test results comparing computation times using HPC versus laptop processing.

5.2 Introduction

Despite the explosion in the use and diversification of research and applications using electroencephalography (EEG), other than one current privately financed project [149; 150], significant recent brain biomarker discovery initiatives have not involved EEG-derived measures [151]. Other neuroimaging techniques, chiefly MRI and fMRI, are the focus of leading initiatives such as the IMAGEN study [152], the U.K. Biobank project [153], the Human Connectome Project [154], the Autism Brain Imaging Data Exchange [155] and the Alzheimer’s Disease Neuroimaging Initiative (ADNI) [156]. Three major factors are hindering the large-scale data collection, analysis, and meta-analysis of EEG data.

Physiologic and experimental variability. The first factor is the native high inter-subject signal variability in EEG signals themselves. Brain anatomic and physiological (as well as cognitive and behavioral) differences between individuals, produce differences in their EEG signals that may severely affect the performance of models and preprocessing techniques meant to generalize across many subjects [151]. Circumventing issues arising from this variability requires major efforts both in data standardization and in data analysis [151; 157–159]. Differences in number and placement of the recording electrodes across studies, and difficulty in co-registering the electrode positions to the head and underlying functional EEG brain sources add further complexity.

Format variability and lack of specificity. A second factor hindering development of large scale EEG data analysis and meta-analysis is the lack of agreement within researchers and system manufacturer communities regarding EEG data formats. Lack of standardization in this area has negatively impacted the growth of EEG data sharing.

Inconsistent and often insufficient documentation of data collection and analysis procedures have made it difficult to reproduce results across laboratories and limited the possibility of bringing together data recorded in different formats using varying experimental paradigms and recording systems and parameters.

The Brain Imaging Data Structure (BIDS) specification initiative is attempting to establish a common set of standards for, first, MRI and fMRI [160] and most recently MEG [161], iEEG [162], and EEG [163] data. The emerging BIDS standards are sets of interrelated research community-developed specifications for organizing, archiving, sharing, and easily analyzing brain imaging data collected within and across studies and laboratories. The new BIDS standards, now being rapidly adopted by neuroinformatics projects in the US and in Europe, address the issue of heterogeneity in the structure of archived data across studies. Importantly for EEG data, the root BIDS standard supports the Hierarchical Event Descriptor (HED) system for annotating the precise nature of experimental events recorded in the data [157], thereby promoting greater depth of detail and improved consistency across labs in the descriptions of experimental events recorded in the data.

Need for high-performance computing. A third factor hindering the development of large-scale EEG analysis is that, given the success of applications of machine learning methods to very large data archives in an increasing number of fields, and the relatively large amount of EEG data that could potentially be made available for such analyses, there is an increasing need for using high-performance computing (HPC) resources. However, there are few readily available tools for solving difficult EEG signal processing problems such as EEG source localization using realistic electrical head models [164–166]; time-frequency (for example, [167–169]) and cross-frequency coupling analysis [17; 26; 117]; stochastic analyses including independent component analysis (ICA) [170; 171]; and a growing variety of machine learning methods (see examples in [172; 173]). Thus, for many EEG researchers lack of access to sufficient well-documented and formatted data, readily

applicable analysis tools, and connected computing resources are major current obstacles to applying new large-scale analysis and meta-analysis methods to EEG and related data

HPC science gateways. Publicly available high-performance computing (HPC) resources do exist at national academic supercomputer centers, but access by neuroscientists to these resources is limited by both administrative and technical barriers including the steep learning curve required to understand HPC hardware, software, usage policies, user environments and to install and run applications on HPC resources [148]. These obstacles are being partially eliminated through the creation of science gateways to publicly-funded HPC resources [174]. Each science gateway provides a customized set of compute-intensive applications to researchers in a specific scientific field, making them available for use via a simple to use web portal and/or an application programming interface (API). These gateways allow scientists to access HPC resources without dealing with the complexities associated with the machine environment. A gateway has been developed for the neuroscience community, the Neuroscience Gateway [175; 176]. Here we present a solution linking the most widely used signal processing environment for EEG research, EEGLAB [57], to the U.S. HPC Extreme Science and Engineering Discovery Environment (XSEDE) computer network (<https://portal.xsede.org/xsede.org>) via the Neuroscience Gateway (NSG).

The Neuroscience Gateway (NSG) [176] is a National Science Foundation (NSF) funded project focused on providing HPC resources to neuroscience researchers. In doing so, NSG aims to lower or eliminate the administrative and technical barriers that currently make it difficult for investigators to use these resources. The NSG (nsgportal.org) gateway offers free computer time to neuroscientists on the XSEDE computer network, providing a ready means to use HPC resources for popular neuroscience tools, pipelines, data processing, and software including NEURON [177], GENESIS [178], MOOSE [179], NEST [180], Brian [181] and PyNN [182], as well as software for analyzing and visualizing structural and functional brain imaging data, such as Freesurfer [183] and MRtrix [184].

NSG leverages access to resources including the supercomputers Comet at the University of California San Diego and Stampede2 at University of Texas Austin, and the Jetstream cloud resource at the University of Indiana.

The EEGLAB signal processing environment. Resources for analyzing EEG data are increasingly freely available as open-source software. Leading examples are the EEGLAB [57], Fieldtrip [30], Brainstorm [31], and MNE [32] tool environments. In 2011, EEGLAB was reported to be the most widely used by the cognitive neuroscience community [33], a report consistent with counts of more recent research publications reporting the use of these respective environments (see Inline Supplementary Figure 1).

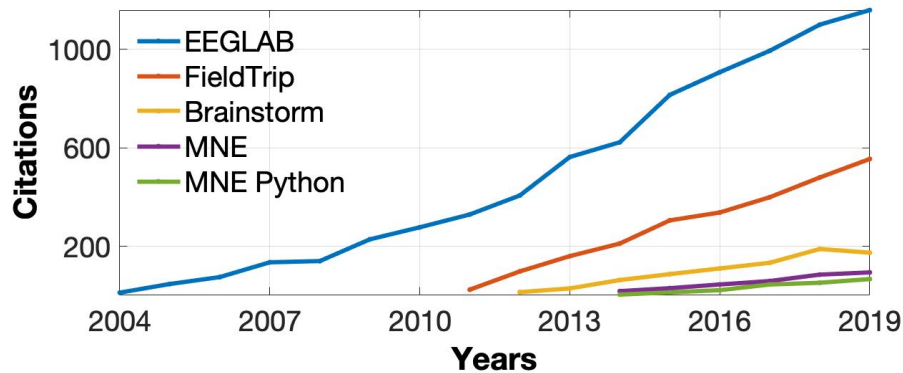


Figure 5.1. *EEGLAB citations.* Citations of the EEGLAB reference paper vs. other open-source EEG analysis software reference paper citations as per the Web of Science (<https://www.webofknowledge.com>, mid-December 2019).

EEGLAB [57], first introduced in 2002 organizing tools first released by Makeig and colleagues at the Salk Institute in 1997, comprises a large set of tools for performing ICA decomposition, time/frequency analysis, and effective source-level (or scalp channel-level) data visualization. EEGLAB combines a simple but powerful graphical user interface (GUI) for data exploration with an easy transition to command line scripting for custom analysis, exploiting the richness of the MATLAB environment (The Mathworks, Inc., Natick, Massachusetts, United States) on which it runs. EEGLAB is also compatible with the open source, cross-platform Octave application, and is readily extensible through its

increasing library of more than 100 plug-in extensions contributed by many laboratories, extensions that appear as easy-to-launch items in the EEGLAB window menu of users who download them. These plug-ins can take advantage of the wide range of easily accessible core MATLAB graphics and other libraries.

Using EEGLAB on high-performance computing resources. In a recent publication [148], we introduced the Open EEGLAB Portal (OEP) software framework enabling interaction between EEGLAB and the Neuroscience Gateway (NSG, nsgportal.org) to offer free use of the high-performance computing (HPC) resources of the U.S. XSEDE network for analysis of EEG and related data from the EEGLAB environment. The OEP makes use of NSG to provide ready access to HPC resources without the burden of complications typically associated with using HPC services.

Earlier, in [148], we previewed the EEGLAB functions supporting the OEP and discussed future development, e.g., a programmatic user interface from EEGLAB to NSG. Here, we introduce the EEGLAB plug-in, *nsgportal* (github.com/scen/nsgportal) that fulfills this purpose – to give cognitive neuroscientists free and easy access to HPC resources from an EEGLAB session running on any lab or personal computer running MATLAB. The current manuscript is organized as follows: In Section 3.5.3, we expand on the structure and capabilities of the OEP, ways of accessing it and the characteristics and structure of the computational jobs. In Section 5.4, we introduce the *nsgportal* plug-in and discuss its architecture, its GUI structure, and its set of MATLAB command line tools. In Section 5.5, we present three *nsgportal* applications, the first illustrating the use of the *nsgportal* GUI, the second its MATLAB command line tools, and the third showing use of these tools to incorporate HPC access into new or existing EEGLAB plug-ins, culminating with building *nsgportal* capabilities into the compute-intensive RELICA (for ‘RELIable ICA’) plug-in of Artoni et al. [170]. Finally, in Sections 5.6 and 5.7, we outline current and future developments and present conclusions from our work to date.

5.3 The Open EEGLAB Portal

5.3.1 NSG and Open EEGLAB Portal structure and capabilities

The Neuroscience Gateway software is based on the Workbench Framework (WF) [185; 186], a software development kit designed to deploy analytical jobs and database searches to a generic set of computational resources and databases. The WF has been used in building successful science gateways such as the CIPRES phylogenetic gateways [185] and the Portal for Petascale Life science Applications and Research (PoPLAR) gateway [187]. Among these, CIPRES is the most successful gateway, handling about 40,000 unique users over a period of ten years (2009 - 2019). This speaks of the maturity of the WF. To create NSG, the CIPRES-WF was modified to hide all the complexities associated with accessing and using HPC resources for various neuroscience tools made optimally available on HPC resources (for more detail, see [188]).

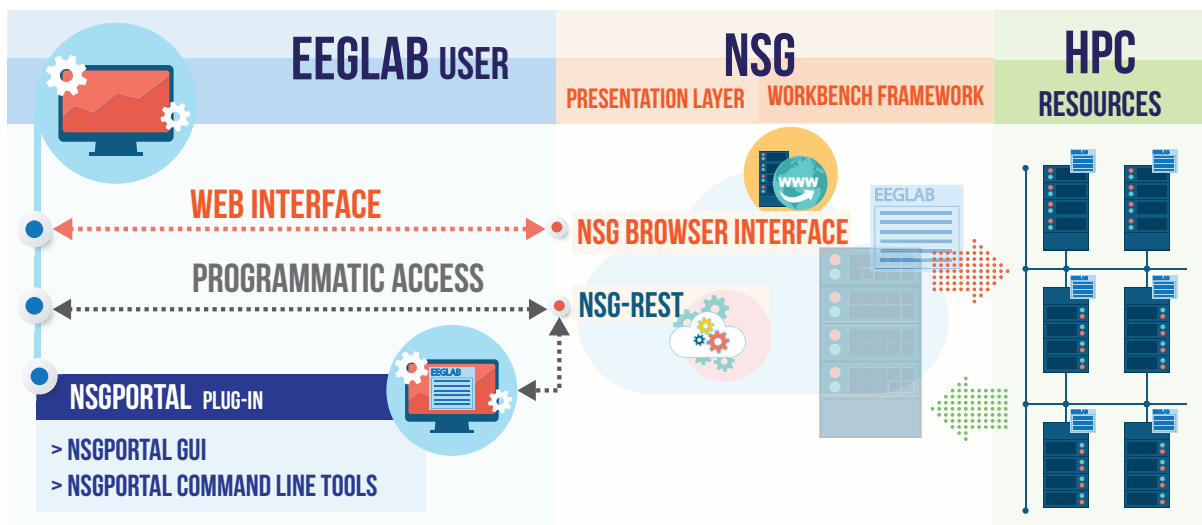


Figure 5.2. *High-level overview of the Open EEGLAB Portal (OEP) workflow.* The OEP architecture has three main components: The EEGLAB user local computer resource (in blue), the Neuroscience Gateway (NSG) (in orange), and the NSG back-end HPC resources (in green). NSG can be accessed via the NSG web interface or through the NSG applications programming interface (API) operating in the Representational State Transfer (REST) environment [189]. The EEGLAB plug-in *nsgportal* exploits the latter access point to provide seamless cross-communication between an EEGLAB session and NSG. Once jobs are submitted to NSG, scripts to execute and data to operate on are deployed to the back-end HPC resources for processing (red arrow). Once job processing is finished, the NSG working directory is first returned to the NSG server (green arrow) and then compressed. Following, the user is notified of the job completion via email. Status of jobs submitted directly from EEGLAB using the API-based *nsgportal* tools can also be monitored using plug-in tools and features. The results from the computation can be retrieved manually via the NSG web interface, or automatically through the *nsgportal* plug-in GUI running within the same or a new EEGLAB session.

The Open EEGLAB Portal (OEP) is built upon the NSG framework, making use of its well-established and tested features, e.g., user interface, account management system, documentation, a ticketing system for bug reports and resolution, a mechanism for user support as well as policies to equitably distribute HPC time. Figure 5.2 (above) presents an overview of OEP workflow. The OEP architecture connects three main components: The user’s local computer, the NSG gateway interface, and the NSG backend HPC resources. NSG resources may be engaged either through the NSG web interface (nsgportal.org) or through its API, NSG-R, operating in the Representational State Transfer (REST)

environment. Here we focus on the interface between EEGLAB and NSG-R through the NSG REST API.

5.3.2 Registering EEGLAB as an NSG application

To use NSG-R, the EEGLAB toolbox itself has been registered to use NSG UMBRELLA authentication, whereby jobs can be submitted to NSG on behalf of multiple registered users. The EEGLAB NSG identifier and key are specified within the *nsgportal* authentication function *pop_nsginfo* (see Figure 5.3) and should not need to be changed by the user.

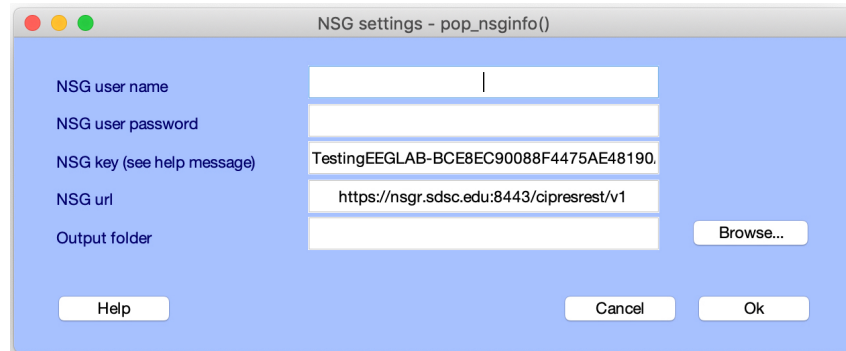


Figure 5.3. *The graphical user interface of `pop_nsginfo.m`. In this window, users can specify their NSG username, password and other options necessary to format the REST API curl commands and store their outputs. The values shown in fields ‘NSG key’ and ‘NSG url’ are set to use EEGLAB by default.*

5.3.3 User interaction with the OEP

The first step to using the Open EEGLAB Portal is to create a user NSG account on the NSG home page (www.nsgportal.org). Obtaining an account first requires providing identity information to allow NSG administrators to verify user authenticity. This information comprises name, email, responsible principal investigator, and institution. Any researcher associated with a not-for-profit institution may apply. Receiving an account credential typically requires two days.

Any job submitted to NSG must be in the form of a compressed *.zip* file of a folder

including raw or pre-processed data files (typically, EEGLAB-formatted), an EEGLAB analysis script to run on MATLAB (The Mathworks, Inc.), plus any custom MATLAB functions not in the main EEGLAB distribution and used in the analysis script. Code for built-in MATLAB and MATLAB toolbox functions, as well as functions in the current EEGLAB release, are already available via NSG and do not have to be provided in the job file.

The version of EEGLAB available in NSG, always the current stable version, includes the most popular EEGLAB data processing plug-in extensions as well as all of the current EEGLAB data import plug-ins. If a toolbox, plug-in, or function used in the job analysis script is not built-in by default in MATLAB, in the EEGLAB release, or in one of the NSG-installed plug-ins, the user must include the corresponding files in the job *.zip* file. (Note: We may add to the installed NSG base other EEGLAB plug-ins of general interest; please email requests to *eeglab@ucsd.edu*).

NSG job submission. User input files can be provided to NSG by using its two main entry ports (see Fig. 5.1): (1) The user can upload through the simple NSG web-based interface (nsgportal.org) a job *.zip* file containing the data and the Matlab code to execute, plus further submission parameters (Section 5.4.4 and 5.4.5). Else, (2) the user can submit, monitor, and retrieve NSG jobs through tools using the NSG REST API (Section 5.4.2).

Job execution. Once the job files and relevant settings are submitted to NSG, independent of the interface used NSG assigns a unique identifier to the job (`Unique_job_identifier`) and generates a job status object with a handle in the following format:

`NGBW-JOB-EEGLAB\_TG-[Unique_job_identifier]`

The NSG front end then sends the input job files with a job request to the remote HPC server for processing. Once the job completes, its working directory is compressed (by

zip) along with the original job input and any output files. Additional files generated by NSG to report HPC scheduler processing (*scheduler_stderr.txt* and *scheduler_stdout.txt*) or job errors (*stderr.txt*), as well as the MATLAB command window output record (*stdout.txt*) are included in this directory. The resulting *.zip* file is transferred to the NSG file system storage [188].

Once the results are ready for retrieval, an email is sent to the user to notify them of the job completion. The user can then download the results contained in the *.zip* output file. Results retrieval must be done through the same NSG access mode used for job submission (the NSG web portal or the NSG REST API).

5.4 The EEGLAB *nsgportal* plug-in

Here we formally introduce *nsgportal*, a plug-in interface between EEGLAB and NSG using the REST API.

5.4.1 Accessing NSG through the plug-in *nsgportal*

Nsgportal capitalizes on the availability of the NSG REST API to provide EEGLAB users an *nsgportal* GUI plus MATLAB command line tools allowing them to manage and interact with NSG jobs directly from the MATLAB and EEGLAB interfaces. (Alternatively, programmatic access using open source curl commands allows the user to perform the same tasks outside the EEGLAB environment, directly from the operating system command line, though doing this requires detailed familiarity with operating system and curl syntax).

5.4.2 Implementation, architecture, and dependencies

The *nsgportal* plug-in, implemented and running in MATLAB, calls curl command line tools that provide an interface to the NSG REST service. As a consequence, to run *nsgportal* the user must have MATLAB and curl/libcurl installed on their computer. Currently, most macOS and Windows (Windows10 and later) releases have curl installed

by default, while Linux users can easily install curl from their software repository. Since MATLAB and curl command line tool code are compatible with macOS, Windows and Linux, the plug-in *nsgportal* will run without modification in MATLAB on any of these platforms. As a plug-in to EEGLAB, *nsgportal* also requires the EEGLAB environment functions to be in the MATLAB path and must be installed as per the standard EEGLAB plug-in installation instructions (available online under scn.ucsd.edu/wiki at *EEGLAB_Extensions_and_plug-ins*).

Functions included in the *nsgportal* plug-in are designated by mimicking the function name hierarchy used in EEGLAB. *Nsgportal* has two main sets of functions, designated by the prefixes *pop_* and *nsg_*. When called with fewer than the required arguments, *pop_* functions open a parameter entry window, else run directly without opening a window. The second class of *nsgportal* functions, with prefix *nsg_*, can be called directly from MATLAB command line or from other MATLAB scripts or functions. These functions perform lower-level tasks than the *pop_* functions. A main plug-in function (*eegplugin_nsgportal*) effects the inclusion and manages the appearance of the *nsgportal* item(s) in the main EEGLAB GUI window menu.

The NSG interface in *nsgportal* is implemented in the functions *nsg_jobs*, *nsg_run* and *nsg_delete*. They each format a curl command as per NSG REST API specifications, as a MATLAB string variable using the inputs received as arguments. The formatted curl command is then issued for execution using the MATLAB system command call. All communication with NSG through the REST interface occurs within an SSL (Secure Sockets Layer) encrypted session, ensuring that the information is transmitted safely. The code samples in Appendix B show typical curl calls to NSG-R in *nsgportal* for B.1 Submitting a job, B.2 Checking job status, B.3 Retrieving job results, and B.4 Deleting a job. Here the submitter must be a registered NSG user (here credited with symbolic credentials *your_username* and *your_password*). These examples do not demonstrate the full scope of the NSG REST API. For more information, see the online

API guide (nsgportal.org/guide.html).

Successful NSG job submission through the REST interface returns a job status *XML* object. In *nsgportal*, this object is parsed into a MATLAB job structure containing job information, parameters, and settings. To manage this interaction, the output from the curl instructions used to submit the job and to request job status are saved in a temporary file with extension *.TXT* or *.XML*. These files are parsed into a structure by *xml2struct.m* within *nsgportal* and are then deleted. Similar workflows apply to other *nsgportal* interactions with NSG (job creation and polling). Details of the *XML* object returned after submitting or managing an NSG job can be found in the NSG REST API user guide (nsgportal.org/guide.html). To download job output results only, the curl command line tool syntax (A.3) is quite similar but there is no need to parse the output *.zip* file, and deleting a job does not return a job object.

Job structures contain information on current job status, results and errors at a more detailed level of specificity and organization. To assist *nsgportal* users in troubleshooting, an error-reporting heuristic has been implemented: this categorizes job status as either *Completed* (job processed), *Processing* (job processing in progress), *MATLAB Error* encountered, or *NSG Error* encountered. A successful NSG job should progress through a series of states (listed in Appendix B), the last (Completed) indicating that the job results are available for download. Note that these four NSG status categories are not the same as the internal *nsgportal* status states, although they are related, as we will show next.

To check NSG job status, *nsgportal* first checks for the latest job message issued by NSG. If it is *Completed*, it checks output file *stderr.txt* to look for error messages issued by MATLAB (typically these are in the first line of the file, enclosed in curly brackets). If an error is found, *nsgportal* sets its internal job status to *MATLAB Error*, or if not, to *Completed*.

If the last message issued for the job is not *Completed*, *nsgportal* checks the *Failed* field in the job structure. If its value is 1, signifying an NSG failure in processing the job,

nsgportal sets its internal job status to *NSG Error*. If not, it sets its internal status to *Processing* to indicate that the job is still being processed. This internal categorization is available only within the *nsgportal* GUI.

5.4.3 Preparing *nsgportal* for use

Installation. All EEGLAB plug-ins including *nsgportal* can be installed by following the procedure described (EEGLAB extensions/plugin-ins under scn.ucsd.edu/wiki/EEGLAB) in the EEGLAB documentation. For the following, we assume the plug-in *nsgportal* has been added to the EEGLAB installation.

Setting NSG credential and other options. Although *nsgportal* uses UMBRELLA authentication within NSG-R, curl commands issued from EEGLAB to interface NSG employ username and password while sharing the common application key ‘EEGLAB_TG’. These commands also use settings that have to be defined by the user, e.g., the NSG URL (typically ‘<https://nsg.sdsc.edu:8443/cipresrest/v1>’). Storing NSG credentials and managing other *nsgportal* settings is performed by *pop_nsginfo*. This function belongs to the group of *pop_* functions introduced in the previous section; we can therefore invoke its GUI by calling the function without any argument (e.g., by simply entering *pop_nsginfo* on the MATLAB command line) or by selecting EEGLAB menu item ‘*Tools > NSG Tools > NSG account info*’ (see Figure 5.3).

The user-provided credentials and options are stored in file *nsg_info.m* in the root of the home folder, or if provided in the path designated for EEGLAB options (see EEGLAB script *eeg_options.m*). Alternatively, the same process of setting up the plug-in, performed in the *pop_nsginfo* GUI (shown in Inline Supplementary Figure 5.3), can be accomplished by calling *pop_nsginfo* from the MATLAB command window and providing the required inputs as key-value pairs, as in the following code sample:

The EEGLAB default *nsgkey* and *nsgurl* values shown in the code sample should not be altered. After registering for and obtaining an NSG account, as described

```

1 pop_nsginfo('nsgusername','[your_username]',
2            'nsgpassword','[your_password]',...
3            'nsgkey','TestingEEGLAB-BCE8EC90088F4475AE48190A1B87EF8D',
4            ...
5            'nsgurl','https://nsgr.sdsc.edu:8443/cipresrest/v1', ...
            'outputfolder','/data/tmp');

```

Listing 5.1. Generic command line call to *pop_nsginfo* to store NSG user credential information

above, the install process described in this section should be performed only once to install the plug-in

5.4.4 The main plug-in GUI: *pop_nsg*

The *nsgportal* GUI. The *nsgportal* graphical user interface (GUI) is controlled by the function *pop_nsg* which, through its associated window and command line interface, is intended to be the primary point of user interaction with NSG within EEGLAB. The main functionalities of *nsgportal*, comprise the submission, managing, deletion, and retrieval of NSG jobs. The functionalities supported by the *pop_nsg* GUI allow users to: (1) Submit an EEGLAB job to NSG and set NSG-R options. (2) Perform test runs of NSG jobs on the local computer (typically using a reduced instruction set). (3) Delete jobs from the user's NSG account. (4) Download NSG job results. (5) Load NSG job results into EEGLAB. (6) Visualize NSG job error and intermediate job logs. (7) Access *pop_nsg* help documentation. The *pop_nsg* GUI (Fig. 5.4) can be invoked by executing '*pop_nsg*' on the MATLAB command windows or by selecting menu item '*Tools > NSG Tools > Manage NSG jobs*' from EEGLAB menu.

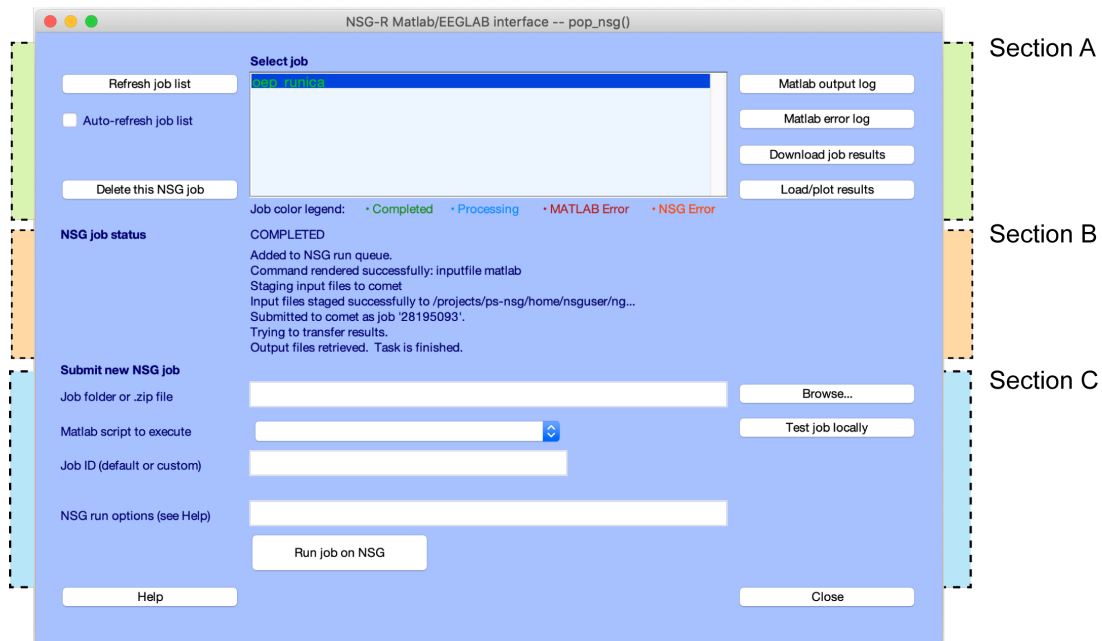


Figure 5.4. The *pop_nsg.m* graphical user interface (GUI) window. This window is intended to be the primary point of user interaction with NSG within EEGLAB graphical environment. From this interface the user can handle the submission, managing, deletion, and retrieval of jobs. The *pop_nsg* GUI comprises three main sections, highlighted by dashed lines in the figure: Top section A (see green backing) allows user to interact with jobs already submitted under their own personal credentials. A color-coded scheme (see its legend in the bottom of the display window) reveals to the user the status of the job selected. Middle section B (orange backing) shows status and other messages associated with the job selected in Section A. Section C (cyan backing) manages user job submission.

The *pop_nsg* GUI has three main sections (highlighted in Fig. 5.4 by backing green, orange and cyan areas; see labels on the upper right side of Fig. 5.4). Top Section A allows the user to interact with their own jobs after submission to NSG. A list of all the active jobs under the user account is displayed in the central text box. There, users can select a job with a mouse click, either to then delete it from the queue, to examine its error logs, or to download and display its results. Here, a font color code with four categories (see display window legend) is used to display job status. Middle Section B displays job status and other messages associated with the job selected in Section A. The first line of this information is one of the job status indicators (Appendix C). The lower Section C allows users to submit jobs to NSG. Here the user can specify the job files (as

a named *.zip* file or folder). The MATLAB data processing script to run must also be defined here, as well as other NSG processing options (e.g., requested maximum run time). Another important feature allows the user to run a (user-prepared) downscaled version of the job on their local computer to test the integrity of the script. To use this option, the user must prepare and identify a limited version of the processing script, for example replacing its major processing loop variable (e.g., `N = all_files_to_process`) with the quicker-running (e.g., `N = 1`). For a detailed explanation of the elements of this GUI and their functionality, see Appendix E.

5.4.5 *Nsgportal* Matlab-interface command line tools

Nsgportal command line tools allow users to largely automate their workflow and make the NSG job submission, job status monitoring, and job results retrieval processes simple for users – and/or EEGLAB functions and plug-ins – to accomplish. Main *nsgportal* functionalities – submitting, monitoring, deleting jobs, and/or retrieving job results without GUI interaction – can be also performed directly using command line calls to function *pop_nsg* providing all its required input arguments. Other useful functions include *pop_nsginfo* (to set up and record user NSG credentials), *nsg_jobs* (to return the current list of user NSG jobs) and *nsg_recurspoll* (to monitor the status of an NSG job). See their function help messages for usage examples.

Setting up user NSG credentials. After installing the *nsgportal* plug-in in EEGLAB, function *pop_nsginfo* can be used to specify the user NSG credential. This function was introduced in Section 5.4.3, where an example of its command line call was presented in the code sample in Listing 5.1.

Submitting and managing NSG jobs. Function *pop_nsg* is the workhorse of the *nsgportal* command line tools. It enables users to: (1) create and run an NSG job (*'run'* option); (2) test a short form of the job on your local computer (*'test'* option); (3) retrieve NSG job results (*'output'* option), and (4) delete an NSG job (*'delete'* option). The function

```

1 [NSGJobStructure, AllNSGJobStructure] = ...
2     pop_nsg('run', 'path/to/my/job/folder', ...
3     'filename', 'my_job_script.m');

```

Listing 5.3. Example command line call running an NSG job from the MATLAB command line using *pop_nsg*

pop_nsg must be called with key-value input pairs as in the following pseudo-code:

```

1 NSGJobStructure = pop_nsg('option_name', 'option_value');

```

Listing 5.2. Generic key-value input pairs used to invoke *pop_nsg*

For options *'run'* and *'test'*, the second argument must always be the pathname of the job *.zip* file or the folder containing the job to be submitted (or locally tested). Using any of these two options also requires a second pair of compulsory arguments (first argument, *'filename'*) to identify the *.m* processing script file to be run in the user test or NSG run . For instance to run a job located in *path/to/my/job/folder* running MATLAB script *my_job_script.m*:

When *pop_nsg* is executed with any of the four mode parameters (*'run'*, *'test'*, *'output'*, or *'delete'*), the function returns two outputs: (1) a job structure containing all relevant information of the submitted or managed job, and (2) a cell array listing all NSG jobs currently in the user NSG account. In the example code above, these two outputs are called *NSGJobStructure* and *AllNSGJobStructure* respectively. These output objects use the same NSG job structure introduced in Section 5.4.2. When *pop_nsg* is called with options *'output'* or *'delete'*, users can pass the job ID, NSG job structure, or the job URL (see Section 5.4.2) as the second argument. To use either of these two options, the referenced NSG job must exist in the user's NSG REST account; to use the *'output'* option, the job must have completed. This flexibility in invoking *pop_nsg* is supported by the *nsgportal* functions *nsg_getjobid* and *nsg_findclinetjoburl*, which respectively translate job URLs to job IDs and vice versa. Function *nsg_jobs* retrieves information about the jobs associated with the user's NSG credential. Users can request a

recursive check of the status of a submitted job using *nsg_recurspoll* (see Appendix B.2). The call below sets the polling interval to 2 minutes (120 seconds).

```
1 NSGJobStructure = nsg_recurspoll('My_Job_ID', 'pollinterval', 120 );
```

Listing 5.4. MATLAB command line call to *nsg_recurspoll*

Recursive polling in *nsg_recurspoll* is implemented using the MATLAB timer class to create an object that manages a recursive execution of the sub-function *nsg_poll*, which in turn executes *nsg_jobs* repeatedly to check job status by pulling the job status object.

5.4.6 *Nsgportal* documentation

A comprehensive *nsgportal* wiki (<https://github.com/sccn/nsgportal/wiki>) linked to the *nsgportal* repository (<https://github.com/sccn/nsgportal>) , as well as function help messages contained in *nsgportal* functions document its use.

The *nsgportal* wiki. The wiki begins with a quick introduction to the Open EEGLAB Portal and details how to register as an NSG user and use its web portal. Details on installation and use of the *nsgportal* plug-in are provided. To represent the workflow within *nsgportal*, we have generated a plug-in function-calling scheme that may ease future modifications and maintenance (<https://github.com/sccn/nsgportal/wiki/Scheme-of-plug-in-functions-call>). Finally, a section of the wiki presents detailed examples of how to use the *nsgportal* plug-in both from the EEGLAB GUI and from the MATLAB command line. Additional examples are given illustrating how to use *nsgportal* command line tools to equip new or existing EEGLAB plug-ins with HPC optional capability.

***Nsgportal* function help messages.** Each main function of *nsgportal* begins with a detailed help message in EEGLAB format: a brief description of the function, a usage example, a table of compulsory and optional input parameters, and a description of the outputs.

5.5 Applications and examples

Below, we show three applications using the *nsgportal* plug-in, applied to a representative EEG dataset. The first two applications illustrate the submission, managing, results retrieval, and deletion of jobs from the *nsgportal* GUI and MATLAB command line, respectively. The third application demonstrates the use of *nsgportal* command line tools to implement a computationally intense EEGLAB plug-in that gives the user the option to perform the computation on HPC resources via NSG. We first demonstrate this capability by implementing a simple plug-in using *nsgportal* functions, and then by extending the more complex RELICA plug-in [170] to include the option to run its computationally demanding processing on NSG resources. Finally, we report the results of testing the NSG-enabled RELICA on actual EEG data.

5.5.1 The test data and simple job example

Test data description and preprocessing. To demonstrate the use of *nsgportal*, we use a single dataset from the EEG/MEG study reported in [143] and made publicly available on *OpenNeuro.org* (accession number: ds000117). For information on the experimental paradigm and a description of the preprocessing steps applied to these data, see Appendix D. The preprocessed data are a set of 132 EEG 3-sec data epochs time-locked to presentations of visual stimuli and recorded from 70 EEG channels at a sampling rate of 250 Hz, saved as an EEGLAB dataset.

The test job script. Electric potentials recorded with EEG are the summation of the underlying activity of multiple brain activity sources. Under favorable circumstances, the activity from these sources can be separated by ICA [167]. However, ICA computation may be lengthy and computationally expensive. Here we show how to build a job script to use NSG to decompose the data into maximally independent components (ICs) by applying Extended Infomax Independent Component Analysis (ICA) using the EEGLAB

```

1 % Launch EEGLAB
2 eeglab;
3 % Load the sample EEGLAB dataset
4 EEG = pop_loadset('wh_sub011_proc.set');
5 % Decompose the data into independent component (IC) processes
6 EEG = pop_runica(EEG,'icatype','runica');
7
8 % Plot the scalp maps of the first (and largest) 20 ICs
9 pop_topoplot(EEG,0,[1:20],'EEG Data epochs',[4 5],0,'electrodes','on
  ');
10 % Save the figure as a JPEG file
11 print('-djpeg','IC_scalp_maps.jpg');
12 % Save the dataset, now including the ICA decomposition matrix
13 pop_saveset(EEG,'filename','wh_sub11_proc_output.set');
14
15 % Delete the input dataset to reduce the output file size
16 delete('wh_sub011_proc.set');
17 delete('wh_sub011_proc.fdt');

```

Listing 5.5. Job script ‘*runica_on_nsg.m*’

function *runica*.

First, we create a job folder (*runica_on_nsg*) containing the test data (files *wh_sub11_preproc.set* and *wh_sub11_preproc.fdt*) plus a simple MATLAB script (*runica_on_nsg.m*) that executes the decomposition via NSG and plots some results. The script (Code sample in listing 5.5 below) loads the dataset, runs the decomposition, plots results, then saves the figure file and output EEGLAB dataset (now including the ICA mixing and unmixing matrices) to return to the user. The input dataset is (optionally) deleted. The figure (*IC_scalp_maps.jpg*) shows scalp maps (scalp projection patterns) of the largest 20 ICs (arranged in a 4-by-5 grid).

Assuming we have obtained an NSG account and have installed the *nsgportal* plug-in, we can then zip the job folder, log in to the NSG website (www.nsgportal.org), and submit the job by identifying the *.zip* file containing the job folder. Else, we can do this and more using the *nsgportal* GUI.

Building a job test script for local execution. A second, ‘test’ version of the job script (*test_runica_on_nsg.m*) in the job folder, differs from *runica_on_nsg.m*

in only one way: it reduces its computational complexity by asking for a much smaller number of ICA training steps (here, it sets the *runica* option '*maxsteps*' to only 5 instead of the default 512). The *nsgportal* GUI can then be used to run the test script on the user's computer to verify the accuracy of the syntax in the rest of the script, thereby avoiding wasting user and NSG time and resources attempting to run scripts that still need debugging. For jobs involving many data files (for different participants and/or conditions) and/or surrogate data analysis, this might also involve severely limiting the number of datasets and/or loop iterations to minimize compute time while allowing MATLAB, running on the user's local computer, to find any coding mistakes before launching the much longer-running full job script on NSG.

5.5.2 Submitting, managing and retrieving the job using the *nsgportal* GUI

After we have installed the *nsgportal* plug-in (Section 5.4.3), we launch the EEGLAB GUI (command, `eeglab`) and then the *pop_nsg* GUI – either by entering *pop_nsg* on the MATLAB command line or by selecting item '*Tools > NSG Tools > Manage NSG jobs*' from the EEGLAB GUI menu. To submit the sample job for processing via NSG from the *nsgportal* GUI (Figure 5.4 above), we first browse and select the (compressed or uncompressed) job file *runica_on_nsg.zip* using the '**Browse**' button. After selecting the job file, the edit box '**Job folder or .zip file**' and the drop-down menu '**Matlab script to execute**' are populated with the path to the zip file and the names of the scripts *runica_on_nsg.m* and *test_runica_on_nsg.m*. The field '' is also populated with a default job name that we choose to change to the more recognizable '*runica_on_nsg_test*'.

Running the test script locally. Field '**Matlab script to execute**' has the two *.m* file entries: *runica_on_nsg.m* and *test_runica_on_nsg.m*. To test the script syntax on our local computer, we first select the test script (*test_runica_on_nsg.m*) and press button '**Test job locally**'. This script is the light version of the job script

runica_on_nsg.m. After the job completes its local processing without any error, we are sure the job script syntax at least contains no syntax errors. Testing the jobs script locally is an optional step not required to submit a job for processing in NSG. However, this check may save a lot of time in troubleshooting syntax errors that may delay and interfere with the completion of the job.

Submitting the job to NSG. After testing the downscaled version of the job locally, we selected the script to execute in NSG (*runica_on_nsg.m*) in the field ‘**Matlab script to execute**’. Next, in edit box ‘**NSG run options**’, we set the job maximum time allocation (here) to one hour by typing ‘*runtime*’, 1. The unit here is hours, and the maximum setting is 48 hours. Note: do not set this to less time than the job will require! Then, we click the button ‘**Run job on NSG**’ to submit the job for processing. Upon successful job submission, the Job ID we assigned (*runica_on_nsg_test*), is shown in the list of job records NSG associates with our NSG user credential. The current status of the job (‘INPUTSTAGING’) is displayed in the job status panel (see Appendix C for status name interpretations).

Monitor job status. After the job is submitted, to ask *pop_nsg* to check periodically for its status check the checkbox ‘**Auto-refresh job list**’. Messages giving the current NSG status of the selected job are then printed (by default, every 30 seconds) on the MATLAB command line. The text color used to display the job ID and job status in the *nsgportal* GUI, will also be updated to show the current job status.

Retrieving the MATLAB command window output of the job. The MATLAB command window output issued in the NSG MATLAB session during the job processing can be checked from *pop_nsg* GUI. To retrieve and display this information in the MATLAB browser, we used the button ‘**Matlab output log**’. It is possible that this information is not available at the time of the request; in this case the user will be notified of this by a MATLAB pop-up window.

Retrieving and loading job results. When the job has completed, an email

```

1 >> ./runica_on_nsg/
2 >>./runica_on_nsg/IC_scalp_maps.jpg
3 >>./runica_on_nsg/runica_on_nsg.m
4 >>./runica_on_nsg/wh_sub11_proc_output.set
5 >>./runica_on_nsg/wh_sub11_proc_output.fdt
6 >>./scheduler_stderr.txt
7 >>./scheduler_stdout.txt
8 >>./stderr.txt
9 >>./stdout.txt
10 >>Done.
11 >>File downloaded and decompressed in the output folder specified in
    the settings
12 >>Accessing job: "https://nsgr.sdsc.edu:8443/cipresrest/v1/job/
    my_username/NGBW-JOB-EEGLAB-TG-9FB3515E932C4CA8AADAEF6D40167F0D"
    on NSG...
13 >>Done.

```

Listing 5.6. The code above shows the MATLAB command window output after retrieving job results using the ‘**Download job results**’ button in the *pop_nsg* GUI. Lines 1-5 show the files generated during the processing. The input job data files were (optionally) removed by our test script, so are not contained in the downloaded results.

to this effect is sent to the registered user. At this point, the job status in the GUI also reads ‘COMPLETED’. We can then download the job results by clicking on ‘**Download job results**’. The names of the downloaded and unzipped files are then printed on the MATLAB command window. The following code sample (Listing 5.6) resembles the message printed on the MATLAB command window after downloading the results of the *runica_on_nsg.zip* test job.

Here, we can see that both results and files submitted are in the downloaded file, saved in the path defined previously in *pop_nsginfo*.

To explore the output files, we use the file explorer implemented in the button **Load/plot** results. This application can load datasets into EEGLAB as well as display a wide range of image formats generated during a job processing in NSG. From this file explorer, we can navigate to (by double-clicking into) the downloaded folder *runica_on_nsg* and access the result files. For instance, after navigating to the folder with the results of the job in this example, if the file *wh_sub11_proc_outpout.set* is selected and opened by


```

1 % The job zip file is located in home directory
2 path2zip = 'Users/eeglabHome/runica_on_nsg.zip';
3 % Execute the job
4 currentjob = pop_nsg('run',path2zip,'filename', 'runica_on_nsg.m', '
    jobid','runica_on_nsg_test');

```

Listing 5.7. Using *pop_nsg* for submitting the *runica_on_nsg.zip* job for processing at NSG

using the button **Load/plot**, the main EEGLAB window will pop up, showing that the selected data has been loaded and that the ICA decomposition has been performed (the ICA weights field in the GUI reads 'yes').

Deleting a job. After retrieving the results, we may proceed to delete the job from our NSG account, here by selecting the job '*runica_on_nsg_test*' in the *pop_nsg* job list and clicking the button '**Delete this NSG job**'. This functionality can be used either on finished jobs or jobs being currently processed. The deletion of jobs from NSG is accomplished by calling *nsg_delete* from *pop_nsg*.

5.5.3 Submitting, managing and retrieving a job using *nsgportal* command line tools.

Next, we show how to create, manage, and retrieve NSG jobs using *nsgportal* command line tools to automate a data analysis pipeline. For this purpose, we will use the same *runica* test job and data introduced in Section 5.5.1. The *nsgportal* command line tools were introduced in Section 5.4.5; there their key components are described in more detail. Here, we again assume we have obtained an NSG account and have installed the *nsgportal* plug-in (see Section 5.4.3).

Submitting a job to NSG. The code sample below uses function *pop_nsg* to submit the job *runica_on_nsg.zip* for processing via NSG. Here we store the path to the job *.zip* file in a variable; this path can also be passed directly as a string to the function:

Notice that three pairs of key-value parameters were used here ('*run*', '*filename*' and '*jobid*'). The '*filename*' argument is compulsory when using the option '*run*' to specify the

top-level script run in the job. Default options in *pop_nsg* assign a randomly generated ID to the job. However, we encourage users to specify a meaningful job ID for their NSG jobs, as this will help in later job management, especially when the number of user submitted jobs is large. In the code sample above, we specify the job ID (*'runica_on_nsg_test'*) using the input parameter *jobid*. After executing the code above, the function *pop_nsg* returns a MATLAB NSG job structure for the submitted job (*currentjob*) that we will use as an input for the next processing steps. Although not shown here, users can also specify other job parameters while submitting a job by providing key-value pair arguments to the function call. These optional arguments include the following: 1) *'outfile'*: the local folder to which the NSG results are to be downloaded. The default value for this parameter is the jobname and job ID plus prefix *'nsgresults_'* (for example, *nsgresults_testjob_1234*). 2) *'runtime'*: the maximum time (in wall clock hours) to allocate for computation of the job on NSG. The default value is 0.5 hours (the maximum value is 48 hours). Note, again, that if the job takes longer to complete than asked for here, it will be stopped! This maximum time estimate is used by the job scheduler to maximize XSEDE resources. *'subdirname'*: Name of the sub-directory containing the script file to run on NSG if the script file is not in the job's root folder. The options available in *pop_nsg* are not restricted to those shown here, as the list may grow based on the users needs. The changes and additions are always made available through the *nsgportal* documentation (Section 5.4.6). **Monitoring job status periodically.** After the job is submitted to NSG it will be processed on the XSEDE network. We can check the status of the job periodically by calling function *nsg_recurspoll*, providing as arguments the job structure obtained when submitting the job (job ID, URL or structure are valid inputs), *currentjob*, and the polling interval in seconds. This function is particularly handy for running functions that operate on the output of a submitted NSG job. Below (Listing 5.8), we specify the job structure *currentjob* as a first argument and 120 seconds (2 minutes) as the polling interval:

```
1 currentjob = nsg_recurspoll(currentjob, 'pollinterval', 120);
```

Listing 5.8. Command line call initiating recursive polling of the job status of job '*currentjob*'.

Once initiated, function *nsg_recurspoll* exits when the NSG job has completed and its output is ready to be accessed by the user. The function also returns a job structure containing the status of the specified job. This structure is basically the same as that provided as input, but with the job status field updated. **Retrieving job results.** After the job completes and function *nsg_recurspoll* exits, job results can be retrieved using *pop_nsg* by providing the job structure following the first argument '*output*'.

```
1 currentjob = nsg_recurspoll(currentjob, 'pollinterval', 120);
```

Listing 5.9. Command line call to *pop_nsg* to retrieve results of job '*currentjob*'.

The input *currentjob* contains the NSG job structure of the job we want to retrieve. As in the example using *nsgportal* GUI, the output files are downloaded to the location defined by the user using *pop_nsginfo* (see Section 5.4.3). **Deleting an NSG job.** Below (Listing 5.10), we use the output structure of *nsg_recurspoll* to delete the job from the NSG record associated with the user's NSG credential.

```
1 NSGjobstruct = pop_nsg('delete', currentjob);
```

Listing 5.10. Command line call to delete the job whose job structure is *currentjob*.

The output here is the NSG job structure (*NSGjobstruct*) of the job deleted. Note that when this command is executed the job is deleted from the user's NSG account and can no longer be accessed.

In this sub-section, we have shown a proof of concept example automating the use of NSG resources using *nsgportal* command line tools. With minor changes to the job script, the example here might be scaled up to process data from many subjects within a single job, etc.

The tools framed here can also be used to implement HPC access offered by NSG from within any EEGLAB function or plug-in. This is the focus of the next section.

5.5.4 Accelerating EEGLAB plug-in performance using *nsgportal* tools

EEGLAB plug-in extensions allow users to build and publish new or customized data processing and visualization functions using EEGLAB data structures, functions, and conventions. Plug-in functions can be called conveniently by selecting the new menu item(s) they introduce into the EEGLAB menu of users who have downloaded and installed them. In this Section, we use *nsgportal* command line tools to implement access to NSG HPC capabilities in: (1) an example plug-in that performs the same processing as in the sample job shown in Section 5.1 and, (2) an example adding *nsgportal* command line calls to an existing plug-in, RELICA [170].

Example plug-in: RUNICA_NSg

Here we describe a simple yet fully functional EEGLAB plug-in (RUNICA_NSg) exemplifying the application of *nsgportal* command line tools to implement HPC computation from within EEGLAB plug-ins. To create this plug-in, we modified the simple *runica_on_nsg.m* script presented above (Section 5.5.1) by turning it into a function *eeg_runica_nsg.m* that accepts inputs that 1) point to the data matrix to decompose, and 2) select the ICA algorithm to use in the decomposition. To allow the sample plug-in to be called directly from the EEGLAB GUI menu, two additional functions are needed: function *eegplugin_runica_nsg* manages the listing of the plug-in in the EEGLAB menu, while *pop_runica_nsg* manages the inputs, first popping up a parameter entry window if its two required input arguments are unspecified, and then redirecting computation either to the local computer resource or to NSG. The example RUNICA_NSg plug-in function code is available online, both from the EEGLAB Extension Manager

(http://scn.ucsd.edu/eeglab/plugin_uploader/plugin_list_all.php) and from its GitHub repository (https://github.com/scn/nsgportal_manuscriptsupport/tree/master/runica_nsg_plugin).

The workflow of RUNICA_NSJ is purposely simple. When its main *pop_* window function *pop_runica_nsg* is invoked either from the EEGLAB menu or MATLAB command line, the data matrix in the currently loaded EEG dataset (*EEG.data*) is selected for decomposition. Two further parameters are required to specify: (1) whether the computation should be performed on the user's computer or on XSEDE network HPC resources via the user's NSJ account, and (2) which ICA algorithm to use (here, given only two choices: '*runica*' and '*jader*'). If the user chooses to perform the computation on their local computer (second argument '*local*') *pop_runica_nsg* will perform the processing locally (by calling *eeg_runica_nsg.m*), returning the input EEG dataset structure, now containing the computed ICA decomposition parameters.

If the user chooses to perform the computation through NSJ (second argument '*nsg*'), the function *pop_runica_nsg* will: (1) create a temporary folder, (2) save the EEG data passed as input in this folder, and (3) generate and save in this folder a MATLAB script (*runica_nsg_job.m*) including a call to the RUNICA_NSJ plug-in function *eeg_runica_nsg* including the user-specified parameters. Then *pop_runica_nsg* will proceed to submit this job folder to NSJ using *nsgportal* command line tools, identifying *runica_nsg_job.m* as the main script to execute (first argument '*filename*'). After submitting the job, *eeg_runica_nsg* will exit, returning the NSJ job ID assigned to the task.

The user can provide this job ID to *pop_nsg* to monitor the job progress and then, when it completes, use it to retrieve its results through *pop_runica_nsg*. The process of job submission and retrieval, as implemented using *nsgportal* command line functions, is shown in Fig. 5.5, in which the *nsgportal* functions used in the process are shown in green boxes.

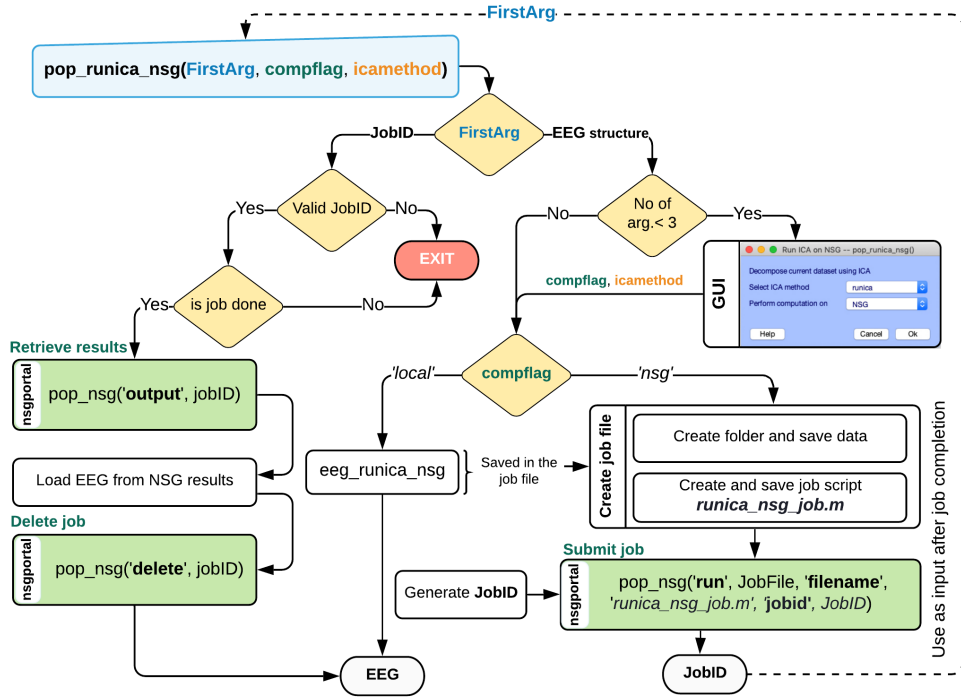


Figure 5.5. Workflow of the example plug-in *RUNICA_NSg*. This plug-in illustrates the use of *nsgportal* command line functions that enable EEGLAB functions and plug-in tools to run on HPC resources via NSG. After installing the plug-in in the `eeglab/plugins/` directory, the user can call it via the EEGLAB menu (*Tools > Compute ICA via NSG*) or equivalently by invoking function *pop_runica_nsg* from the MATLAB command window. In the second case, an EEG data structure must be provided as input. If invoked from the menu, or when either of its other two arguments, *compflag* (run locally or via NSG) or *icamethod* (ICA algorithm to use) are not specified, a MATLAB window will pop up to allow the user to input these parameters. When *pop_runica_nsg* is invoked from the EEGLAB GUI, the currently loaded EEGLAB dataset (*EEG.data*) is processed and the pop window is always launched to allow manual entry of arguments *compflag* and *icamethod*. When the computation is to be run via NSG, *pop_runica_nsg* creates a temporary folder and saves the current EEG dataset in it along with function script *eeg_runica_nsg.m* and a custom script (*runica_nsg_job.m*) generated to invoke this function using the parameters provided by the user (here, *icamethod*). This script, along with the data saved, is then submitted to NSG for processing under a user-designated job ID (*JobID* in the figure). Function *pop_nsg* with options ‘run’, ‘filename’ and ‘jobid’ performs these steps. Following job submission, *pop_runica_nsg* exits, providing the submitted *jobID* as output. Once the NSG job is complete (as flagged by an email message sent to the user from NSG, or via the open *pop_nsg* GUI), the user must retrieve its results by calling *pop_runica_nsg* with the *JobID* as its first and only argument. Internally, *pop_runica_nsg* will check the validity of the job ID and will then call *pop_nsg* (with arguments ‘run’ and ‘delete’) to retrieve the results and then delete the job from the user’s NSG account record. Finally, the EEG data structure provided originally as input, including the ICA decomposition parameters, is loaded into EEGLAB by *pop_runica_nsg* and returned as output.

Adding HPC computation to the RELICA plug-in

RELICA (for ‘*RE*Liab*le* ICA’) is a method that uses a stochastic approach to characterize the reliability of component processes identified by ICA decomposition of a multichannel dataset, by analyzing the stability of the component processes under bootstrap resampling. RELICA thus performs multiple ICA decompositions of bootstrapped versions of the input data, then clusters the ICs from all the decompositions based on similarities between their time courses. By default, the number of clusters corresponds to the number of ICs per decomposition. Each cluster in the analysis is associated with an exemplar IC, selected to be the IC nearest to the cluster centroid in the clustering space. Finally, RELICA computes a measure of the compactness of each bootstrap IC cluster that provides a within-subject measure of IC stability for each IC.

Figure 5.6 shows a high-level view of the process implemented in the original RELICA implementation by Artoni et al. [170]. Most of the computation time here is spent in the sequence of ICA decompositions performed on the bootstrapped data (red area, Fig. 5.6) . The high computational load required to perform these processes, particularly for large datasets with many channels, makes the RELICA plug-in a good candidate for running on NSG high-performance computing resources using *nsgportal* tools.

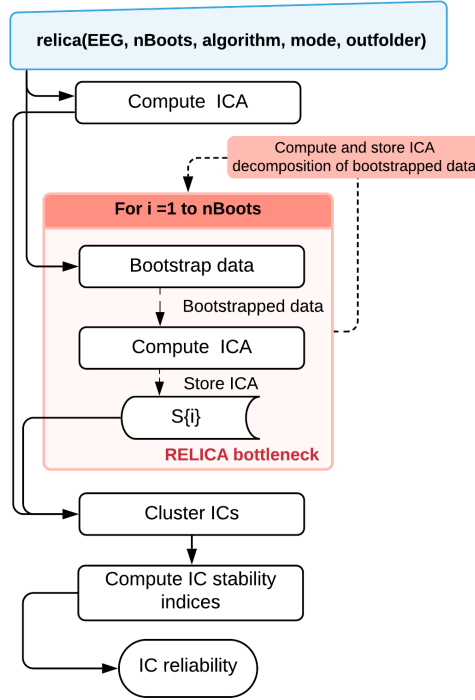


Figure 5.6. *Original RELICA workflow.* A high-level view of the original implementation of RELICA as proposed by Artoni et al. [170]. Given an EEG dataset, *relica* first performs a reference decomposition of the whole input data, and then multiple ICA decompositions of bootstrapped versions of the same data (red area). The function *relica* then clusters the identified ICs from all the decompositions and from this computes a measure of the stability of each IC in the reference decomposition.

Here we modified the RELICA implementation to reduce its runtime. First we parallelized the main for loop in the RELICA bottleneck (red area) using *parfor* from the MATLAB Parallel Computing Toolbox to speed processing on either local (multi-core) or (NSG-accessed) HPC computers. Then we added *nsgportal* code to enable it to run on XSEDE HPC resources via NSG. Fig. 5.7 shows a high level view of the resulting NSG-capable RELICA code.

The user first provides an EEGLAB dataset (in the form of an EEG data structure) for processing by *relica*. As in Section 5.5.4, by specifying the ‘*nsgflag*’ argument as ‘*local*’ or ‘*NSG*’, the user can direct the computation to the user’s local computer or to NSG resources. In the latter case, *relica* will create a command line call script to invoke *relica*

on the HPC resource and pass the user-selected options to it. This script plus the input data are saved in a temporary folder and submitted to NSG as a job using *pop_nsg* with argument '*run*'. If the submission is successful, the (assigned or optionally user-specified) job ID is returned as a text output. This job ID must later be used to retrieve the results when available. Here, the user provides *relica* with the job ID as first argument.

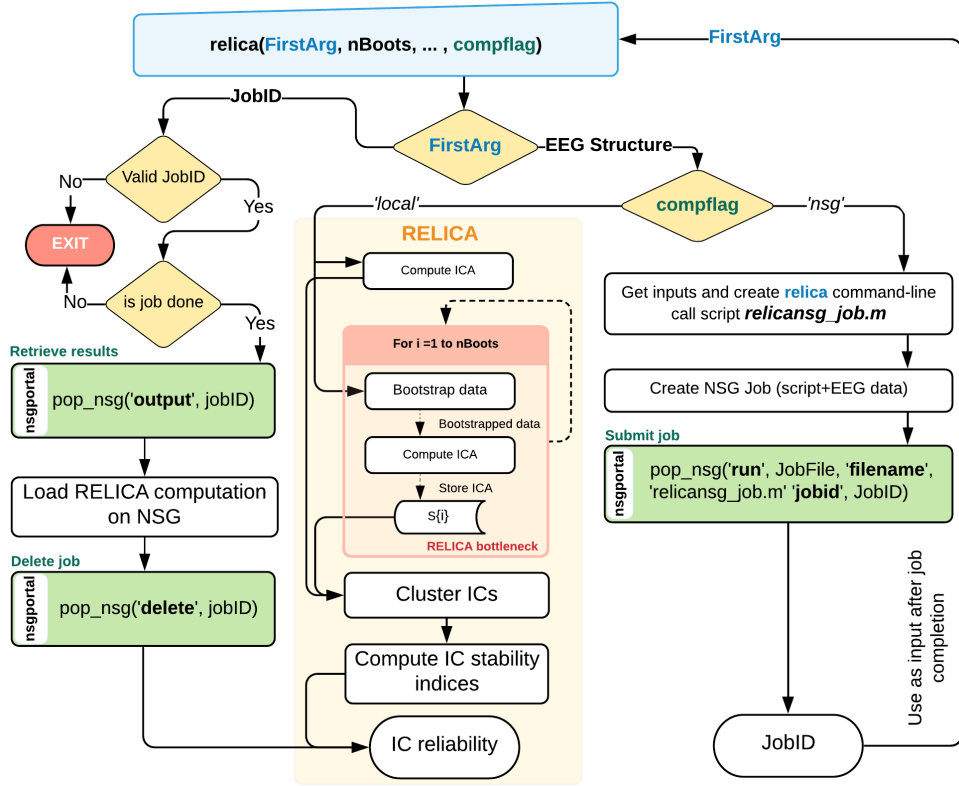


Figure 5.7. *The NSG-capable RELICA plug-in.* A high level view of the NSG-capable RELICA plug-in. Here we have added a last input, *compflag*, to the original inputs of the main function *relica* to indicate where the computation should be carried out (*'nsg'* or *'local'*). When *relica* is called with an EEG set structure and *compflag* is *'local'*, computation is performed as usual (central yellow panel). If *compflag* is *'nsg'*, *nsgportal* code in *relica* first saves the current EEG dataset in a temporary folder also containing a script (*relica_nsg_job.m*) generated to run *relica* on NSG with the parameters provided by the user. This script and the data provided as input are submitted to NSG for processing using *pop_nsg* with first argument *'run'* (green panel on right). Additional inputs are provided to indicate the script to run (*relica_nsg_job.m*), and the designated job ID. This uses *pop_nsg* options *'run'*, *'filename'*, and *'jobid'*. After the job submission, *relica* exits, providing the *jobID* as output. After this, the plug-in processing proceeds as in *runica_nsg* (Figure 5.5). Once the NSG job is complete, the user provides the job ID to NSG to retrieve the results by again calling *relica* with the job ID as the first and only argument. After checking the validity of the submitted job ID, *relica* uses *pop_nsg* (with arguments *'run'* and *'delete'*) to retrieve the results and then delete the job from the user's NSG account. Finally the submitted EEG dataset plus the RELICA-computed IC reliability measures are provided in the function output.

When *relica* confirms the validity of the provided job ID and that the job is complete, it downloads the job results (*pop_nsg 'output'*), deletes the job from the user's NSG account

(*pop_nsg* ‘*delete*’) and loads the output of *relica* into the EEG dataset structure. The results of RELICA are stored in the EEG dataset structure, allowing seamless integration of NSG computation into RELICA post-processing and results visualization.

RELICA performance: HPC versus local computing

To test the performance of RELICA running via NSG on XSEDE cluster Comet, we ran RELICA on the sample EEG dataset referenced in Section 5.5.1, setting the number of bootstraps to 100 and specifying ICA algorithm ‘*runica*’ (Extended Infomax ICA) [190; 191]. The two bars on the left in Fig. 5.8 show RELICA processing time on a modern laptop (MacBook Pro, Intel(R) Core(TM) i7-4770HQ 4-core CPU @ 2.20 GHz) and on NSG (1 24-core node of XSEDE network HPC resource Comet). Timestamps for each NSG process were retrieved from the job status XML object introduced in Section 5.4.2. Despite the relative simplicity of the task, via NSG the RELICA job was performed 11 times faster than on the laptop. A detail of the NSG processing time, magnified x10 in the gray panel on the right side of Fig. 5.8, shows the time the job spent in the different NSG job processing stages including data handling and job scheduling. The scheduling delay depends on the queue wait time of the HPC machine to which NSG sends jobs for processing, so may vary with machine demand intensity.

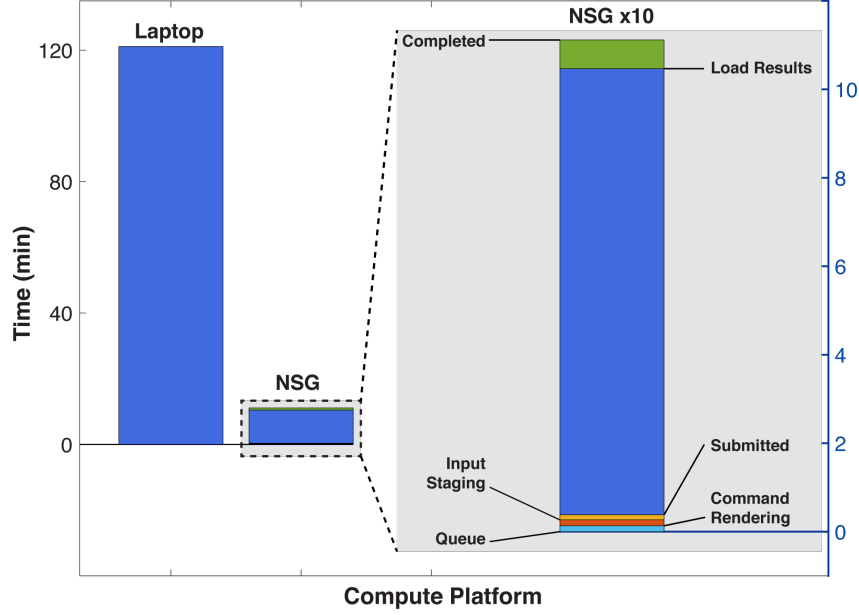


Figure 5.8. *RELICA* processing time on NSG versus a laptop. The two leftmost bars represent recorded processing time running RELICA on the sample data using a laptop (MacBoook Pro, Intel(R) Core(TM) i7-4770HQ 4-core CPU @ 2.20GHz) and via NSG (1 node, all 24 cores of the XSEDE network cluster Comet) using 100 iterations of ICA decomposition using ‘*runica*’ [171]. A detail of NSG processing time magnified by 10 is shown in the right gray panel. Here the time has been split to show the time the job spent on each of the NSG stages. Here the initialization stages (bottom of right bar) produced only a small delay, though this may vary with current system load. Data upload and download times (not shown) here required about as much time as the 3 job initialization stages (right column bottom), but these delays are still negligible compared to the time saved by performing the computation using NSG versus on the laptop. Upload and download times depend heavily on the Web bandwidth and traffic load between the user machine and UCSD.

In a second test, we explored the reliability of the RELICA computation performed via NSG vis a vis two trial runs performed on the same laptop. To do this, we computed pairwise correlations between the IC cluster exemplar maps obtained from RELICA computed via NSG and from the two runs on the laptop using relic options as in Section 5.5.4. Results of this analysis are shown in Fig. 5.9, in which the sorted best-matching IC-pair correlations between the cluster exemplar IC maps obtained from the NSG and first laptop run (NSG, $Laptop_1$), NSG, and the second laptop run (NSG, $Laptop_2$), and between the two laptop runs ($Laptop_1$, $Laptop_2$), are shown in blue, green and orange

traces respectively. As we can see, in all the output IC-pair comparisons (NSG versus $Laptop_1$, NSG versus $Laptop_2$, and $Laptop_1$ versus $Laptop_2$) at least 60 of the 70 IC pairs are near-perfectly replicated. The small remaining variability results from the stochasticity of the ICA computation itself. Fig. 5.9 also shows four sets of best-matching IC maps from the three runs. Black lines show the mean correlation-sorted rank of the three correlated IC cluster exemplar maps. Numbers above the maps give the stability index reported by RELICA for these ICs. The high similarity of the maps (even for the rightmost pair), demonstrate the high level of consistency of RELICA output across runs and computing resources.

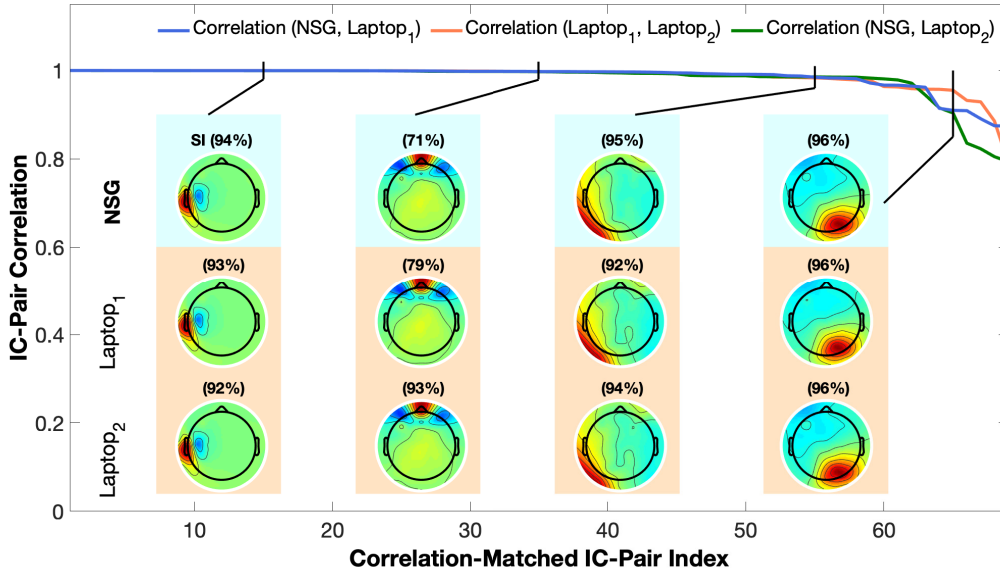


Figure 5.9. *Cluster exemplar scalp map correlations* for best-matching IC pairs in the output of the same RELICA decomposition run (twice) on a laptop and (once) on XSEDE network cluster Comet via the Neuroscience Gateway using *nsgportal*. Top blue trace: correlations of best-matching IC scalp map pairs, sorted by decreasing correlation. Scalp maps: Scalp maps of some best-matching ICs from the three decompositions. Black traces point to the mean correlation rank of these ICs in the three pairwise comparisons among the three maps. Numbers above the scalp maps give the stability index reported by RELICA for these ICs.

5.6 Discussion

Recently [148] we introduced the Open EEGLAB Portal (OEP) framework providing free use of high-performance computing (HPC) resources from the EEGLAB environment through the Neuroscience Gateway (NSG, *nsgportal.org*). Here, we report the release of an EEGLAB plug-in, *nsgportal*, implementing an interface between NSG and EEGLAB running on MATLAB on any Web-connected computer. The *nsgportal* plug-in uses the capabilities of the NSG REST API to enable optional HPC processing of computationally intensive tasks directly from any internet-connected EEGLAB GUI, or from any suitably adapted EEGLAB function or the MATLAB command line. The *nsgportal* features a flexible and user friendly GUI that allows users to directly submit, manage and retrieve jobs running on the U.S XSEDE network of HPC resources from within any EEGLAB session. The set of *nsgportal* command line tools supporting the functions of its GUI, enables users to create their own HPC computation-enabled functions, function pipelines, and EEGLAB plug-ins, while making the NSG job submission and results retrieval process as simple as possible. Detailed documentation and wiki pages are available to support *nsgportal* use and maintenance.

Here we have described *nsgportal* functions and interfaces, and pointed to online documentation to support their use, maintenance and further development. We have presented three examples to illustrate the use of *nsgportal* and its potential applications. The first two examples demonstrated how to use *nsgportal* to create, manage, retrieve results from, and then delete an NSG job within the *nsgportal* GUI and from the EEGLAB session command line. Here we used a simple script to compute an ICA decomposition of an EEG dataset, save the results, and plot images of the resulting IC scalp maps. The structured approach and guidelines used in these examples can be easily modified to accomplish more computationally intensive tasks.

We then demonstrated (Section 5.5.4), through a concrete example, how to use

nsgportal command line functions to implement direct HPC access from within any EEGLAB function or plug-in. For this purpose, we modified the job presented in Section 5.5.1 by turning the earlier example script into a function accepting inputs pointing to the data matrix to decompose and selecting the ICA algorithm to use in the decomposition. The resulting simple example plug-in (RUNICA_NSJ) is available from the EEGLAB plug-in manager and includes abundant comments and documentation to support implementing HPC access using *nsgportal* within other EEGLAB functions and plug-ins. Please note that EEGLAB plug-ins can be run in NSJ processing scripts without including any *nsgportal* functions; many commonly-used plug-ins are available in the EEGLAB NSJ distribution – any that are not yet available directly within NSJ can be uploaded to NSJ, along with the data and any other custom EEGLAB functions. Using the *nsgportal* functions within EEGLAB functions and plug-ins (as in Sections 5.5.4 and 5.5.4) is a convenient option for authors of new or existing plug-ins to consider, as using *nsgportal* functions they can provide users an NSJ-mediated computation option within the plug-in itself.

To show this application operating in an existing EEGLAB plug-in in, we chose to use the more computationally demanding RELICA plug-in of Artoni [170]. The addition of *nsgportal* tools to RELICA was presented in Section 5.5.4.

To explore the impact of these changes on RELICA performance, we ran a RELICA task on NSJ and on a modern laptop (MacBoook Pro, Intel(R) Core(TM) i7-4770HQ 4-core CPU @ 2.20GHz). The NSJ computation was eleven times faster than on the laptop, and the resulting output across three runs (one via NSJ and two on the laptop) were practically indistinguishable (Section 5.5.4). NSJ computation time included data handling and scheduling delays. These times can vary depending on the system load at the time of submission. To probe variability in the times required for NSJ data handling and job scheduling, we submitted the ICA decomposition job (Section 5.5.1) to NSJ repeatedly over a month, and recorded computation times and scheduling delays. Total computation time averaged 31 ± 3 min (mean ± standard deviation). Of this, job scheduling accounted

for 2–4 min (with a non-normal distribution). These results provide a rough idea of the stability of NSG regarding scheduling delay, here under relatively modest user demand.

Job file upload and download durations may depend heavily on user Internet bandwidth and job size. Data upload and download times for the RELICA test job were negligible compared to the required length of the computation. However, we must warn users that data I/O speed may be an important factor to consider when weighing the advantages of running NSG jobs requiring very large data file uploads. NSG data I/O speeds vary greatly depending on the bandwidth and then-existing traffic load of each link in the user connection to UCSD.

Lack of adequate computational power currently acts as a kind of ‘compute horizon’ boundary on scientific imagination itself – enhancing computational power can thus, at its best, both prompt and enable new scientific vision and discovery. However, before submitting EEGLAB jobs to run via NSG using *nsgportal*, we encourage users to consider the time cost/benefit ratio of submitting a job for processing to NSG versus using a local compute resource. There may be instances in which the use of a local resource and/or alternate analysis method are more efficient than using NSG – for example, in cases in which uploading the data to NSG might itself require more time than performing the computation locally or where the applied algorithm may itself be easily optimized to achieve the desired computation speed. Use of the substantial compute resources available via NSG should be exercised thoughtfully, particularly in these times in which reducing the human carbon footprint appears key to preserving human civilization in its current form.

Another important aspect to consider before submitting a job to NSG is whether and to what extent the task itself can take good advantage of parallel computing resources. In some (extreme) cases, the overhead involved in setting up and maintaining parallel processing might require more time than the actual computation. Jobs running iterative processes each requiring a long computation time (such as optimal electrical head modeling

or ICA decomposition of large datasets), or jobs applying intensive processing to a large amount of data (applying time/frequency analysis to a large EEG study, for example) may best take advantage of NSG resources. Other EEGLAB scripts, functions, and pipelines that use vectorized code already optimized for local computation may not benefit from being sent to NSG for batch processing. The latter could only be justified when the data handled by the pipeline are too large for a personal computer to manage. Finally, jobs that require human interaction (e.g., manual data cleaning) cannot take useful advantage of batch-mode HPC processing.

Some current limitations: Currently, on the Comet supercomputer running (at present) 24-core nodes (note: the planned successor to Comet will use 64-core nodes), and NSG can assign a maximum of one node (24 cores) to each EEGLAB job, though this limitation will soon be relaxed with the planned introduction on Comet of the MATLAB Parallel ServerTM library. Another current limitation is the lack of NSG support for data retention and shareability within user NSG accounts. Under a current NeuroElectroMagnetic data Archive and tools Resource (NEMAR) project operating under the OpenNeuro data archive umbrella (openneuro.org), we are developing a solution that will ultimately allow joint use within NSG of personal and/or public data published on OpenNeuro.

5.7 Conclusions

Here we have described *nsgportal*, an EEGLAB plug-in toolbox providing neuroscientists and others using the EEGLAB signal processing environment (sccn.ucsd.edu/eeglab) free access to HPC resources of the U.S. XSEDE high-performance computing network directly from the EEGLAB GUI or session MATLAB command line through a programmatic interface to the Neuroscience Gateway (nsgportal.org). We also showed how to use the *nsgportal* command line tools to equip EEGLAB functions and plug-in toolboxes with

direct HPC run capabilities. Finally, we performed a proof-of-concept test of *nsgportal* performance on a suitable EEG data processing task, and confirmed the equivalence of the obtained results.

The ease and flexibility of the *nsgportal* tools can accelerate any neuroelectromagnetic data research in need of higher computational power. In particular, dynamic modeling and machine learning approaches that are now being applied to human electrophysiological data can require unprecedented computing power. Limitations in available computing resources might be said to constitute a ‘compute horizon’ beyond which researchers cannot or dare not explore or imagine. Extending the compute horizon for electrophysiological imaging research using *nsgportal* tools may hopefully stimulate researchers to use new research methods built on richer data models and thereby obtain new results stimulating new insights into brain function and health.

5.8 Acknowledgements

Chapter 5 is a reprint of the material as it appear in Ramon Martinez-Cancino, Arnaud Delorme, Dung Truong, Fiorenzo Artoni, Kenneth Kreutz-Delgado, Subhashini Sivagnanam, Kenneth Yoshimoto, Amitava Majumdar, and Scott Makeig. *The Open EEGLAB Portal Interface: High-Performance Computing with EEGLAB*. NeuroImage (2020): 116778. The dissertation author was the primary investigator and author of this paper.

This work was supported by the National Institutes of Health, U.S.A. (R01-NS047293, R24-MH120037, R01-MH123231) and the National Science Foundation (1458840), and by a gift from The Swartz Foundation (Old Field, NY).

Figure 5.2 uses elements designed by macrovector/FreePik and FreePik (*freepik.com*). The authors also thank all *nsgportal* beta users for their encouragement and valuable feedback and to Dr. Johanna Wagner for helping with the references.

Chapter 6

Conclusion

It is now widely acknowledged that neural rhythms at different frequencies can temporally interact within and between brain areas. The contributions of this dissertation directly answer a growing interest in studying these spatiotemporal interactions and their relationship to normal and pathological states. Among these interactions, the event in which the amplitude of high-frequency cortical field activity is locked to changes in the phase of a slower brain rhythm is known as phase-amplitude coupling (PAC). The PAC phenomenon has received a heightened interest due to several reports of its role in fundamental brain functions and its links to various brain pathologies.

There is an apparent consensus that to disentangle the role(s) of PAC in neurophysiology, the question of *when* it occurs is as central as *where* or *how*. However, although PAC analysis has been approached through multiple methods and theoretical frameworks, only a few have addressed the transient character of this process. Among these, none has reported details of PAC dynamics while incorporating its possible directional delay characteristics. In this dissertation, I addressed these problems in Chapters 2 and 3 by approaching the problem of PAC estimation through the information theory framework and, specifically, through the use of its local measures [100].

In Chapter 2, I proposed and validated a new method for computing time-resolved PAC using local mutual information, MIPAC. MIPAC comprises two variants for its

application to continuous and event-related data. Validation of the new method was carried out using simulated and actual human electrocorticography (ECoG) data that displayed high PAC as confirmed by other well established PAC methods [26–28; 52].

In Chapter 3, I explored the potential use of a directed local information theory measure, local transfer entropy, to estimate the delays and dynamics of the PAC process. A viable PAC estimation methodology was evaluated in simulated and actual ECoG data from a recorded human epileptic seizure. Here I reported initial indications that local transfer entropy can be used to detect the onset and offset of modulation process periods revealed by mutual information phase-amplitude coupling. In this chapter, I also discussed several problems that must be addressed before presenting a new PAC estimation method using local transfer entropy.

Another PAC-related problem addressed in this dissertation is the availability of dedicated and optimized computational tools for studying the PAC phenomena. In line with the current interest in studying PAC, most of the major academic software for analyzing electrophysiological data (e.g., Fieldtrip [30], Brainstorm [31], MNE [32]) have implemented at least one PAC estimation method. In Chapter 4, I led the EEGLAB software efforts and proposed *PACTools*: an EEGLAB plug-in running on MATLAB (The Mathworks, Inc.) that computes PAC in either continuous or event-related data. *PACTools* comprises five methods for PAC estimation: Mean Vector Length Modulation Index [28], Kullback-Leibler Modulation Index [26], Phase-Locking Value [42], General Linear Model Modulation Index [27], and the newly developed Mutual Information PAC (MIPAC). *PACTools* works seamlessly with EEGLAB structures and functions, and features a parallelized code for faster computation. Since the computation of time-resolved PAC may be computationally expensive, especially if lengthy continuous data is used, I implemented built-in access to high-performance computing resources at the Neurosciences Gateway (*nsgportal.org*) into *PACTools*. To support this feature, I developed the EEGLAB plug-in *nsgportal*, which works within the framework of the Open EEGLAB Portal [148], and

interfaces EEGLAB with NSG directly while running in MATLAB on any personal lab computer. The *nsportal* plug-in was introduced in Chapter 5, where its architecture and implementation were discussed along with examples for its application.

This dissertation should be regarded as composed of two parts, one comprised of Chapters 2 and 3, that addresses the theoretical development of methods for PAC estimation and evaluation, and another section, composed by Chapters 4 and 5, supporting the methods proposed by creating tools and functionalities that facilitate their use. These two parts contribute directly to the dissertation’s primary goal (see Section 1.4) and constitute a theoretical and analytical framework for approaching the study of PAC through its dynamical features.

The work presented here is far from being finished, and the limitations associated with each of the contributions are discussed in the conclusions of each of the Chapters 2, 3, 4 and 5. In particular, the role of the information transfer in the PAC phenomena in Chapter 3, and the extension of *PACTools* functionalities to address the computation and analysis of PAC in data from multiple subjects remains unaddressed. This will be the focus of future studies and developments.

Given the relevance of the current work to address dynamics in the PAC phenomena and having provided the computational tools to carry such analysis, I am very optimistic that this dissertation’s contributions may clear the way and catalyze the efforts to better understand PAC’s physiological role in the healthy and pathological brain.

Appendix A

Phase-amplitude coupled signal modeling

A.1 Phase-amplitude coupled signal modeling

Let us start by considering a carrier signal S_c of frequency f_c and amplitude A_c as in Eq. A.1.

$$S_c(t) = A_c \sin(2\pi f_c t) \quad (\text{A.1})$$

The amplitude of this carrier wave will be modulated by a signal S_m of frequency f_m and amplitude A_m (Eq. A.3).

$$S_m(t) = A_m \cos(2\pi f_m t) \quad (\text{A.2})$$

Then, the simulated PAC signal is expressed as the sum of the AM and modulator signal A.3. Where the modulation is controlled by the parameter $M(t)$.

$$S_{sim}(t) = [1 + M(t)S_m(t)]S_c(t) + S_m(t) \quad (\text{A.3})$$

In the simulations presented in this manuscript, $A_c = 5$, $f_c = 40$ and $f_m = 5$, the time series of the modulation strength $M(t)$ were constructed according to the type of

simulation. For this, the signal was split in five identical segments each of 1 s duration. Three different functions were used to generate the $M(t)$ time series simulating dynamic changes in the amplitude modulation. In the first two cases (on/off) boxcar and linearly ramped time courses were used to simulate coupling in alternate signal segments. In the third case, coupling was applied as the absolute value of a sinusoid across all the segments. The scripts and functions used to simulate the PAC signals in the manuscript can be obtained from the authors upon request.

Appendix B

NSG REST API commands in *nsgportal*

The code samples below show typical curl calls to the NSG REST API used in *nsgportal* for: submitting a job, checking job status, retrieving job results, and deleting a job. Lines in the code samples have been numbered for tutorial purposes (they are not present in the actual code). These examples assume the user has been registered in NSG and obtained a valid username and password, designated as *your_username* and *your_password* respectively. The application key and NSG URL (*nsg.sdsc.edu:8443/cipresrest/v1*) are designated as *\$KEY* and *\$URL* respectively.

```
1  curl -s -u your_username:your_password \  
2      -H cipres-appkey:$KEY \  
3      $URL/job/your_username \  
4      -F tool=EEGLAB_TG \  
5      -F input.infile_=@"/data/TestingEEGLABNSG.zip" \  
6      -F vparam.filename_=run_ica_nsg.m \  
7      -F metadata.clientJobId=TestingEEGLABNSG \  
8      -F vparam.outputfilename_="nsgresults_TestingEEGLABNSG" \  
9      -F vparam.number_nodes_=1 \
```

Listing B.1. Code sample B.1

B.1 Submitting a job

The code sample above (Code sample B.1) shows a `curl` command following NSG REST API specification for submitting a job to NSG using EEGLAB as a tool (see Line 4). The job zip file is specified with the parameter `input.infile_` in Line 5, and the script to run with the parameter `vparam.filename_` in Line 6. A job ID is indicated with the parameter `metadata.clientJobId` in Line 7. Although not compulsory in NSG REST API specifications, in *nsgportal* the use of a job ID is mandatory. The rationale for this is to provide meaningful job identifiers to users, rather than a long job url used in NSG to tag and track jobs. A default job ID assigned in *nsgportal* is created by taking the name of the file in `input.infile_` and attaching three randomly generated numbers in its end. This job ID can be modified at the user's convenience. In the same way, in *nsgportal*, the name of the output file is assigned a default value created by attaching the prefix `'nsgresults_'` to the value defined in `metadata.clientJobId`. This value is set in Line 8 of the code sample above by using the parameter `vparam.outputfilename_`. Finally, we define the number of nodes requested for computation with the parameter `number_nodes_` in Line 9. At this moment, NSG only allows the selection of one node (currently, of 24 cores) for each EEGLAB job. For more details on the MATLAB call of the code above, see *nsgportal* function *nsg_run*.

B.2 Checking job status

```
1 curl -s -u your_username:your_password \
2     -H cipres-appkey:$KEY \
3     $URL/job/your_username
```

Listing B.2. Code sample B.2

The code above (Code sample B.2) allows checking for the status of all jobs under the user `myusername` account. The MATLAB call for this code, as well as the one to

retrieve job results, presented next in Appendix B.3, are implemented in the *ngportal* function *nsg_jobs*.

B.3 Retrieving job results

```
1 curl -s -u your_username:your_password \  
2     -H cipres-appkey:$KEY \  
3     $URL/job/your_username/NGBW-JOB-EEGLAB-TG-Unique_job_identifier/  
    output
```

Listing B.3. Code sample B.3

The code sample above (Code sample B.4) is used to retrieve the job files and results (with job url:

`$URL/job/nsgusername/NGBW-JOB-EEGLAB-TG-Unique_job_identifier )`

B.4 Deleting a job

```
1 curl -u your_username:your_password \  
2     -H cipres-appkey:$KEY \  
3     -X DELETE \  
4     $URL/your_username/NGBW-JOB-EEGLAB-TG-Unique_job_identifier
```

Listing B.4. Code sample B.4

The code above (Code sample B.4) is used to delete the job specified by the job URL in line 4 from the user's NSG account. The MATLAB call for this code is implemented in the *ngportal* function *nsg_delete*.

Appendix C

NSG jobs status list

A successful NSG job should progress through the following states (NSG):

- **QUEUE** - The job has been validated and placed in the NSG queue.
- **COMMANDRENDERING** - The job has reached the head of the queue, and NSG has created the command line that will run the job on the HPC resource.
- **INPUTSTAGING** - NSG has created a temporary working directory for the job on the execution HPC host and copied the input files over.
- **SUBMITTED** - The job has been submitted to the HPC host.
- **LOAD_RESULTS** - The job has finished running on the HPC host, and NSG has begun to transfer the results back.
- **COMPLETED** - Results have been successfully transferred and are available in the NSG web portal.

Appendix D

Wakeman-Henson data processing

The stimuli presented in this experiment comprised two sets of 300 grayscale photographs, half from known people and half from unknown people. Images of known, unknown and scrambled faces were presented for random durations between 800 and 1,000 ms following the appearance of a fixation cross cue with a random duration between 400 and 600 ms. During the 1,700-ms interstimulus interval, a central white circle was presented. The participant was told to fixate centrally throughout the experiment and asked to press one of two keys based on the degree of bilateral symmetry (more or less) of each presented face. EEG data were recorded synchronously with MEG data using an Elekta Neuromag Vectorview 306 system (Helsinki, FI). A 70-channel Easycap (easycap.de/wordpress) cap was used to record the EEG data, with electrode layout conforming to the extended 10–10% system. The EEG electrodes location was digitized by using a 3-D digitizer (Fastrak Polhemus Inc., Colchester, VA, USA). Data were acquired at an 1100-Hz sampling rate with a low pass filter applied below 350 Hz and no high pass filter. The EEG reference electrode was placed on the nose, and the common ground electrode was placed at the left collar bone. Stimuli were presented during six, 7.5-min runs. The protocol is described in more detail in Wakeman and Henson, 2015 [143]. Data preprocessing, performed using EEGLAB and custom scripts, written in MATLAB (The Mathworks, Inc.), proceed through the following steps:

1. Data from subject sub-11 was downloaded from OpenNeuro at https://openneuro.org/crn/datasets/ds000117/snapshots/1.0.3/files/sub-08:ses-meg:meg:sub-08_ses-meg_task-facerecognition_run-01_meg.fif
2. Raw EEG data and event information were imported using FileIO plug-in for EEGLAB. The location of the fiducials were added to the channel location and the channel montage was rotated to match EEGLAB format. Event latencies were corrected by 34 ms as reported by the data authors. Finally, the six sequentially collected task run files were merged and saved.
3. Downsample data from 1100 Hz to 500 Hz using EEGLAB function *pop_resample*.
4. High-pass filter the data above 1 Hz (FIR, Hamming windowed, transition bandwidth 1 Hz).
5. Remove line noise (50 Hz) by applying a Hamming-windowed (sinc) FIR notch filters centered at the line frequency and its harmonics (e.g., 50 Hz, 100 Hz, 150 Hz, and 200 Hz).
6. Remove segments in the data containing non-brain artifact (e.g., high-frequency muscle noise and other irregular artifacts) as identified by visual inspection.
7. Apply common-average reference computation involving all channels
8. Extract data segments time-locked to the presentation of each of the three main events (*known*, *unknown* and *scrambled face*) and spanning from 1 sec before to 2 sec following stimulus onsets.
9. Save the dataset in EEGLAB *.set* format.

Code for the processing described here can be accessed in the following GitHub repository: https://github.com/sccn/nsqportal_manuscriptsupport.

Appendix E

Nsgportal GUI elements

This appendix describes in more detail the elements and section of the *nsgportal* main GUI in Figure 5.4

Section A: Interacting with submitted NSG jobs.

This section contains the following components:

1. (List box) **Select job:** lists all existing NSG jobs under the user credentials. Font color coding is used to indicate the job state: completed (**Completed**, in green), still being processed (**Processing**, in blue), returning a MATLAB syntax-related job error (**MATLAB error**, in red) or an NSG-related error (**NSG error**, in orange). For example, in Figure 5.4, the job named *oep_runica* is shown in green, indicating that the job has been completed. Most of the functionalities accessed in this section of the graphical interface will act on the job selected in this list.
2. (Button) **Refresh job list:** refreshes the list of existing NSG jobs under the user account.
3. (Checkbox) **Auto-refresh job list:** flags the interface to update the user NSG jobs status every thirty seconds. The status of the selected job is shown to the right of ‘**NSG job status**’ label. This functionality uses a MATLAB timer object to schedule a sequential update of the status of the jobs.

4. (Button) **Delete this NSG job:** removes the selected job from the user's NSG account.
5. (Button) **Matlab output log:** downloads and displays the accumulated MATLAB command line output for the selected NSG job.
6. (Button) **Matlab error log:** downloads and displays the MATLAB error log for the selected job (if available).
7. (Button) **Download job results:** download the job result *.zip* file from the currently selected job.
8. (Button) **Load/plot results:** launch a window controlling the browsing, loading, and displaying of job results.

Section B: Checking NSG job status.

This section of the GUI window (beneath '**NSG job status**') displays the job status and the messages issued by NSG during the submission and processing of the job currently selected in the jobs list. The job-status is displayed in capital letters and reflects the current stage of the job at the time of the last update (see Appendix C for the list of states). These states, while related to the four color-coded categories in the job list, are mostly referring to more fundamental NSG processing stages in the selected job.

Section C: Submitting a job to NSG.

The lower section of the GUI (beneath '**Submit new NSG job**') is dedicated to submitting jobs to NSG as well as for performing testing of the jobs in the user's local computer. In what follows, we describe and explain the functionality of each component.

1. (Button) **Browse** and (edit box) **Job folder or zip file:** This button launches a window to browse for a *.zip* file or a folder containing the job to be submitted to NSG (or tested locally). Once the file/folder is selected, its full path is displayed in

(edit box) **Job folder or zip file.** The value in the edit box is used in the NSG-R command line option `input.infile_` when submitting a job.

2. (Drop-down menu) **Matlab script to execute:** Show all the MATLAB script files (*.m*) in the job file (*.zip* file or file folder) selected above. The Matlab script for NSG execution or testing must be selected here. In the case when a folder is selected, the listing of files is done by scanning the folder content with the MATLAB function `dir`. When a *.zip* file is selected, the listing of the *.m* files is done through the *nsgportal* function *listzipcontents.m*. This function uses Java objects and methods native to MATLAB to read the content of the *.zip* file without unzipping it. Both options, for folder and *.zip* files, are implemented in the callback of (button) **Browse** in *pop_nsg.m*. The value in this menu is used in the NSG-R command line option `input.filename_` when submitting a job (see examples in Appendix B).
3. (Button) **Test job locally:** runs the job indicated by the job file and MATLAB script selected above on the local computer where *nsgportal* is being executed. Local testing of the jobs is intended to be done by using a downscaled version of the job to be submitted. This can be done, e.g., by using a scaled-down script mimicking the main steps to be performed at NSG, but on a lower scale. Take the example of a job that processes hundreds, say $N=200$, of subjects in a for loop, the downscaled version of this job may result from the shortening of the loop from 200 to 2. This button is strategically located to the right of (edit box) **Matlab script to execute**, to emphasize the idea that job testing should be done on a different script from the one intended to run on the HPC resource via NSG.
4. (Edit box) **Job ID (default or custom):** displays the unique identifier for the NSG job. The information in this field is generated automatically when a job folder or *.zip* file is selected in the **Browse** button above. The default value assigned here is created by taking the name of the job folder or *.zip* file and attaching three randomly

generated numbers at the end. We encourage users to modify this field for easier recall and reference. The value here is used in the optional NSG-R command line option `metadata.clientJobId` when submitting a job. This option is compulsory when the job is submitted from *nsgportal*, since the job ID value is used instead of the long job URL for job management and tracking in the *nsgportal* GUI machinery.

5. (Edit box) **NSG run options (see Help)**: allows the user to specify options for the to be submitted job. The options defined here are appended to the curl command when the job is submitted to NSG. Currently, *nsgportal* includes options for defining the maximum runtime allocation in NSG (*runtime*), the number of NSG nodes to use in the computation (*nnode*) (currently, 1), and other options (see the *pop_nsg* and *nsg_run* help messages). The list of options will eventually include all NSG-R options.
6. (Button) **Run job on NSG**: submits the job to NSG. The button callback calls *nsg_run* and passes the inputs provided by the user in the GUI. These inputs are formatted in a curl command (see the example in Appendix B), before being issued for execution on the local computer.
7. (Buttons) **Help** and **Close** respectively display the help message of *pop_nsg* in a browser window and close the *pop_nsg* GUI.

Bibliography

- [1] Eric R Kandel, James H Schwartz, Jessell Thomas M, Siegelbaum Steven A, and Hudspeth A J. *Principles of neural science*. McGraw-Hill Education, 5 edition, 2012.
- [2] Mental health.
- [3] Take action for the sustainable development goals – united nations sustainable development.
- [4] Adriano B. L. Tort, Robert W. Komorowski, Joseph R. Manns, Nancy J. Kopell, and Howard Eichenbaum. Theta–gamma coupling increases during the learning of item–context associations. *Proceedings of the National Academy of Sciences*, 106(49):20942–20947, December 2009.
- [5] Eelke Spaak, Mathilde Bonnefond, Alexander Maier, David A Leopold, and Ole Jensen. Layer-specific entrainment of gamma-band neural activity by the alpha rhythm in monkey visual cortex. *Current Biology*, 22(24):2313–2318, 2012.
- [6] Ning Cheng, Qun Li, Sitong Wang, Rubin Wang, and Tao Zhang. Permutation mutual information: A novel approach for measuring neuronal phase-amplitude coupling. *Brain topography*, pages 1–16, 2017.
- [7] Roberto C Sotero and Ramón Martínez-Cancino. Dynamical mean field model of a neural-glial mass. *Neural computation*, 22(4):969–997, 2010.
- [8] Brain facts: A primer on the brain and nervous system. 2018.
- [9] György Buzsáki and Andreas Draguhn. Neuronal oscillations in cortical networks. *science*, 304(5679):1926–1929, 2004.
- [10] Gyorgy Buzsaki. *Rhythms of the Brain*. Oxford University Press, 2006.
- [11] David Millett. Hans berger: From psychic energy to the eeg. *Perspectives in biology and medicine*, 44(4):522–542, 2001.
- [12] Hans Berger. Über das elektrenkephalogramm des menschen. *European Archives of Psychiatry and Clinical Neuroscience*, 87(1):527–570, 1929.
- [13] György Buzsáki, Nikos Logothetis, and Wolf Singer. Scaling brain size, keeping timing: evolutionary preservation of brain rhythms. *Neuron*, 80(3):751–764, 2013.
- [14] Fernando Lopes da Silva. Neural mechanisms underlying brain waves: from neu-

- ral membranes to networks. *Electroencephalography and clinical neurophysiology*, 79(2):81–93, 1991.
- [15] Viktor Jirsa and Viktor Müller. Cross-frequency coupling in real and virtual brain networks. *Frontiers in computational neuroscience*, 7:78, 2013.
 - [16] György Buzsáki and Brendon O Watson. Brain rhythms and neural syntax: implications for efficient coding of cognitive content and neuropsychiatric disease. *Dialogues in clinical neuroscience*, 14(4):345, 2012.
 - [17] Ramón Martínez-Cancino, Joseph Heng, Arnaud Delorme, Ken Kreutz-Delgado, Roberto C Sotero, and Scott Makeig. Measuring transient phase-amplitude coupling using local mutual information. *NeuroImage*, 185:361–378, 2019.
 - [18] Yutaka Nonoda, Makoto Miyakoshi, Alejandro Ojeda, Scott Makeig, Csaba Juhász, Sandeep Sood, and Eishi Asano. Interictal high-frequency oscillations generated by seizure onset and eloquent areas may be differentially coupled with different slow waves. *Clinical Neurophysiology*, 127(6):2489–2499, 2016.
 - [19] Jon López-Azcárate, Mikel Tainta, María C. Rodríguez-Oroz, Miguel Valencia, Rafael González, Jorge Guridi, Jorge Iriarte, José A. Obeso, Julio Artieda, and Manuel Alegre. Coupling between beta and high-frequency activity in the human subthalamic nucleus may be a pathophysiological mechanism in Parkinson’s disease. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 30(19):6667–6677, May 2010.
 - [20] Coralie De Hemptinne, Elena S Ryapolova-Webb, Ellen L Air, Paul A Garcia, Kai J Miller, Jeffrey G Ojemann, Jill L Ostrem, Nicholas B Galifianakis, and Philip A Starr. Exaggerated phase–amplitude coupling in the primary motor cortex in parkinson disease. *Proceedings of the National Academy of Sciences*, 110(12):4780–4785, 2013.
 - [21] Romain Goutagny, Ning Gu, Chelsea Cavanagh, Jesse Jackson, Jean-Guy Chabot, Rémi Quirion, Slavica Krantic, and Sylvain Williams. Alterations in hippocampal network oscillations and theta–gamma coupling arise before $\alpha\beta$ overproduction in a mouse model of alzheimer’s disease. *European Journal of Neuroscience*, 37(12):1896–1902, 2013.
 - [22] Elena A. Allen, Jingyu Liu, Kent A. Kiehl, Joel Gelernter, Godfrey D. Pearlson, Nora I. Perrone-Bizzozero, and Vince D. Calhoun. Components of cross-frequency modulation in health and disease. *Frontiers in Systems Neuroscience*, 5:59, 2011.
 - [23] Tyler B Grove, Carly A Lasagna, Ramón Martínez-Cancino, Preetha Pamidighantam, Patricia J Deldin, and Ivy F Tso. Neural oscillatory abnormalities during gaze processing in schizophrenia: Evidence of reduced theta phase consistency and inter-

areal theta-gamma coupling. *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*, 2020.

- [24] Ali Bahramisharif, Ali Mazaheri, Nina Levar, Peter Richard Schuurman, Martijn Figee, and Damiaan Denys. Deep brain stimulation diminishes cross-frequency coupling in obsessive-compulsive disorder. *Biological psychiatry*, 80(7):e57–e58, 2016.
- [25] Stavros I. Dimitriadis, Nikolaos A. Laskaris, Malamati P. Bitzidou, Ioannis Tarnanas, and Magda N. Tsolaki. A novel biomarker of amnesic MCI based on dynamic cross-frequency coupling patterns during cognitive brain responses. *Frontiers in Neuroscience*, 9, 2015.
- [26] Adriano BL Tort, Robert Komorowski, Howard Eichenbaum, and Nancy Kopell. Measuring phase-amplitude coupling between neuronal oscillations of different frequencies. *Journal of neurophysiology*, 104(2):1195–1210, 2010.
- [27] WD Penny, E Duzel, KJ Miller, and JG Ojemann. Testing for nested oscillation. *Journal of neuroscience methods*, 174(1):50–61, 2008.
- [28] Ryan T Canolty, Erik Edwards, Sarang S Dalal, Maryam Soltani, Srikantan S Nagarajan, Heidi E Kirsch, Mitchel S Berger, Nicholas M Barbaro, and Robert T Knight. High gamma power is phase-locked to theta oscillations in human neocortex. *science*, 313(5793):1626–1628, 2006.
- [29] Juhan Aru, Jaan Aru, Viola Priesemann, Michael Wibral, Luiz Lana, Gordon Pipa, Wolf Singer, and Raul Vicente. Untangling cross-frequency coupling in neuroscience. *Current opinion in neurobiology*, 31:51–61, 2015.
- [30] Robert Oostenveld, Pascal Fries, Eric Maris, and Jan-Mathijs Schoffelen. Fieldtrip: open source software for advanced analysis of meg, eeg, and invasive electrophysiological data. *Computational intelligence and neuroscience*, 2011, 2011.
- [31] François Tadel, Sylvain Baillet, John C Mosher, Dimitrios Pantazis, and Richard M Leahy. Brainstorm: a user-friendly application for meg/eeg analysis. *Computational intelligence and neuroscience*, 2011, 2011.
- [32] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A Engemann, Daniel Strohmeier, Christian Brodbeck, Lauri Parkkonen, and Matti S Hämäläinen. Mne software for processing meg and eeg data. *Neuroimage*, 86:446–460, 2014.
- [33] Michael Hanke and Yaroslav O Halchenko. Neuroscience runs on gnu/linux. *Frontiers in neuroinformatics*, 5:8, 2011.
- [34] Ramón Martínez-Cancino, Arnaud Delorme, Dung Truong, Fiorenzo Artoni, Kenneth

- Kreutz-Delgado, Subhashini Sivagnanam, Kenneth Yoshimoto, Amitava Majumdar, and Scott Makeig. The open eeglab portal interface: High-performance computing with eeglab. *NeuroImage*, page 116778, 2020.
- [35] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.
 - [36] Warren S McCulloch. An upper bound on the informational capacity of a synapse. In *Proceedings of the 1952 ACM national meeting (Pittsburgh)*, pages 113–117, 1952.
 - [37] Fred Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183, 1954.
 - [38] Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1:217–234, 1961.
 - [39] Michael Wibral, Raul Vicente, and Joseph T Lizier. *Directed information measures in neuroscience*. Springer, 2014.
 - [40] Thomas Schreiber. Measuring information transfer. *Physical review letters*, 85(2):461, 2000.
 - [41] Michael Wibral, Nicolae Pampu, Viola Priesemann, Felix Siebenhühner, Hannes Seiwert, Michael Lindner, Joseph T Lizier, and Raul Vicente. Measuring information-transfer delays. *PloS one*, 8(2):e55809, 2013.
 - [42] Jean-Philippe Lachaux, Eugenio Rodriguez, Jacques Martinerie, and Francisco J Varela. Measuring phase synchrony in brain signals. *Human brain mapping*, 8(4):194–208, 1999.
 - [43] Roberto C Sotero. Topology, cross-frequency, and same-frequency band interactions shape the generation of phase-amplitude coupling in a neural mass model of a cortical column. *PLoS Comput Biol*, 12(11):e1005180, 2016.
 - [44] Craig G. Richter, Mariana Babo-Rebelo, Denis Schwartz, and Catherine Tallon-Baudry. Phase-amplitude coupling at the organism level: The amplitude of spontaneous alpha rhythm fluctuations varies with the phase of the infra-slow gastric basal rhythm. *NeuroImage*, 146:951–958, February 2017.
 - [45] Michael X Cohen, Nikolai Axmacher, Doris Lenartz, Christian E. Elger, Volker Sturm, and Thomas E. Schlaepfer. Good Vibrations: Cross-frequency Coupling in the Human Nucleus Accumbens during Reward Processing. *Journal of Cognitive Neuroscience*, 21(5):875–889, August 2009.

- [46] Bradley Voytek, Ryan T Canolty, Avgusta Shestyuk, Nathan Crone, Josef Parvizi, and Robert T Knight. Shifts in gamma phase–amplitude coupling frequency from theta to alpha over posterior cortex during visual tasks. *Frontiers in human neuroscience*, 4:191, 2010.
- [47] Takufumi Yanagisawa, Okito Yamashita, Masayuki Hirata, Haruhiko Kishima, Youichi Saitoh, Tetsu Goto, Toshiki Yoshimine, and Yukiyasu Kamitani. Regulation of motor representation by phase-amplitude coupling in the sensorimotor cortex. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 32(44):15467–15475, October 2012.
- [48] Sara M. Szczepanski, Nathan E. Crone, Rachel A. Kuperman, Kurtis I. Augustine, Josef Parvizi, and Robert T. Knight. Dynamic Changes in Phase-Amplitude Coupling Facilitate Spatial Attention Control in Fronto-Parietal Cortex. *PLOS Biology*, 12(8):e1001936, August 2014.
- [49] Esther Florin and Sylvain Baillet. The brain’s resting-state activity is shaped by synchronized cross-frequency coupling of neural oscillations. *NeuroImage*, 111:26–35, 2015.
- [50] Roberto C Sotero, Aleksandra Bortel, Shmuel Naaman, Victor M Mocanu, Pascal Kropf, Martin Y Villeneuve, and Amir Shmuel. Laminar distribution of phase-amplitude coupling of spontaneous current sources and sinks. *Frontiers in neuroscience*, 9:454–454, 2015.
- [51] Jose M Hurtado, Leonid L Rubchinsky, and Karen A Sigvardt. Statistical method for detection of phase-locking episodes in neural oscillations. *Journal of neurophysiology*, 91(4):1883–1898, 2004.
- [52] Bradley Voytek, Mark D’Esposito, Nathan Crone, and Robert T. Knight. A Method for Event-related Phase/Amplitude Coupling. *NeuroImage*, 64:416–424, January 2013.
- [53] Michael X Cohen. Assessing transient cross-frequency coupling in eeg data. *Journal of neuroscience methods*, 168(2):494–499, 2008.
- [54] Kai J. Miller, Gerwin Schalk, Dora Hermes, Jeffrey G. Ojemann, and Rajesh P. N. Rao. Spontaneous Decoding of the Timing and Content of Human Object Perception from Cortical Surface Recordings Reveals Complementary Information in the Event-Related Potential and Broadband Spectral Change. *PLOS Computational Biology*, 12(1):e1004660, January 2016.
- [55] Dora Hermes, Kai J Miller, Herke Jan Noordmans, Mariska J Vansteensel, and Nick F Ramsey. Automated electrocorticographic electrode localization on individually

- rendered brain surfaces. *Journal of neuroscience methods*, 185(2):293–298, 2010.
- [56] Anders M Dale, Bruce Fischl, and Martin I Sereno. Cortical surface-based analysis: I. segmentation and surface reconstruction. *Neuroimage*, 9(2):179–194, 1999.
 - [57] Arnaud Delorme and Scott Makeig. Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis. *Journal of neuroscience methods*, 134(1):9–21, 2004.
 - [58] Andreas Bruns and Reinhard Eckhorn. Task-related coupling from high-to low-frequency signals among visual cortical areas in human subdural recordings. *International journal of psychophysiology*, 51(2):97–116, 2004.
 - [59] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.
 - [60] David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
 - [61] Robert M. Fano. *Transmission of information. A Statistical Theory of Communication*. The MIT Press, 1963.
 - [62] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.
 - [63] Donald Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pages 268–275. Association for Computational Linguistics, 1990.
 - [64] Robin AA Ince, Bruno L Giordano, Christoph Kayser, Guillaume A Rousselet, Joachim Gross, and Philippe G Schyns. A statistical framework for neuroimaging data analysis based on mutual information estimated via a gaussian copula. *Human brain mapping*, 38(3):1541–1573, 2017.
 - [65] Michael Wibral, Joseph T Lizier, and Viola Priesemann. Bits from brains for biologically inspired computing. *Frontiers in Robotics and AI*, 2:5, 2015.
 - [66] Alessandro Treves and Stefano Panzeri. The upward bias in measures of information derived from limited data samples. *Neural Computation*, 7(2):399–407, 1995.
 - [67] Jonathan D Victor. Binless strategies for estimation of information from neural data. *Physical Review E*, 66(5):051903, 2002.
 - [68] Stefano Panzeri, Riccardo Senatore, Marcelo A Montemurro, and Rasmus S Petersen.

- Correcting for the sampling bias problem in spike train information measures. *Journal of neurophysiology*, 98(3):1064–1072, 2007.
- [69] LF Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.
 - [70] Germán Gómez-Herrero, Wei Wu, Kalle Rutanen, Miguel C Soriano, Gordon Pipa, and Raul Vicente. Assessing coupling dynamics from an ensemble of time series. *Entropy*, 17(4):1958–1970, 2015.
 - [71] Philipp Berens et al. Circstat: a matlab toolbox for circular statistics. *J Stat Softw*, 31(10):1–21, 2009.
 - [72] Biyu J He, John M Zempel, Abraham Z Snyder, and Marcus E Raichle. The temporal structures and functional significance of scale-free brain activity. *Neuron*, 66(3):353–369, 2010.
 - [73] Scott Makeig. Auditory event-related dynamics of the eeg spectrum and effects of exposure to tones. *Electroencephalography and clinical neurophysiology*, 86(4):283–293, 1993.
 - [74] Ole Jensen, Eelke Spaak, and Hyojin Park. Discriminating valid from spurious indices of phase-amplitude coupling. *Eneuro*, 3(6):ENEURO–0334, 2016.
 - [75] David M. Groppe, Thomas P. Urbach, and Marta Kutas. Mass univariate analysis of event-related brain potentials/fields I: A critical tutorial review. *Psychophysiology*, 48(12):1711–1725, December 2011.
 - [76] Nikolaus Kriegeskorte, W Kyle Simmons, Patrick SF Bellgowan, and Chris I Baker. Circular analysis in systems neuroscience: the dangers of double dipping. *Nature neuroscience*, 12(5):535, 2009.
 - [77] G. McCarthy, A. Puce, J. C. Gore, and T. Allison. Face-specific processing in the human fusiform gyrus. *Journal of Cognitive Neuroscience*, 9(5):605–610, 1997.
 - [78] Richard A Davis, Keh-Shin Lii, and Dimitris N Politis. Spectral analysis for harmonizable processes. In *Selected Works of Murray Rosenblatt*, pages 420–460. Springer, 2011.
 - [79] Peter Grassberger. An optimized box-assisted algorithm for fractal dimensions. *Physics letters A*, 148(1-2):63–68, 1990.
 - [80] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on*

Mathematical Software (TOMS), 3(3):209–226, 1977.

- [81] Anatol Bragin, Gábor Jandó, Zoltán Nádasdy, Jámille Hetke, K Wise, and Gy Buzsáki. Gamma (40-100 hz) oscillation in the hippocampus of the behaving rat. *Journal of Neuroscience*, 15(1):47–60, 1995.
- [82] Laura Lee Colgin, Tobias Denninger, Marianne Fyhn, Torkel Hafting, Tora Bonnevie, Ole Jensen, May-Britt Moser, and Edvard I Moser. Frequency of gamma oscillations routes flow of information in the hippocampus. *Nature*, 462(7271):353, 2009.
- [83] John E Lisman and Ole Jensen. The theta-gamma neural code. *Neuron*, 77(6):1002–1016, 2013.
- [84] Kei M Igarashi, Li Lu, Laura L Colgin, May-Britt Moser, and Edvard I Moser. Coordination of entorhinal–hippocampal ensemble activity during associative learning. *Nature*, 510(7503):143, 2014.
- [85] Nikolai Axmacher, Melanie M Henseler, Ole Jensen, Ilona Weinreich, Christian E Elger, and Juergen Fell. Cross-frequency coupling supports multi-item working memory in the human hippocampus. *Proceedings of the National Academy of Sciences*, page 200911531, 2010.
- [86] Daria Osipova, Dora Hermes, and Ole Jensen. Gamma power is phase-locked to posterior alpha activity. *PloS one*, 3(12):e3990, 2008.
- [87] Brett L Foster and Josef Parvizi. Resting oscillations and cross-frequency coupling in the human posteromedial cortex. *Neuroimage*, 60(1):384–391, 2012.
- [88] Ali Bahramisharif, Marcel AJ van Gerven, Erik J Aarnoutse, Manuel R Mercier, Theodore H Schwartz, John J Foxe, Nick F Ramsey, and Ole Jensen. Propagating neocortical gamma bursts are coordinated by traveling alpha waves. *Journal of Neuroscience*, 33(48):18849–18854, 2013.
- [89] Frédéric Roux, Michael Wibral, Wolf Singer, Jaan Aru, and Peter J Uhlhaas. The phase of thalamic alpha activity modulates cortical gamma-band activity: evidence from resting-state meg recordings. *Journal of Neuroscience*, 33(45):17827–17835, 2013.
- [90] Tom Dupré La Tour, Lucille Tallot, Laetitia Grabot, Valérie Doyère, Virginie Van Wassenhove, Yves Grenier, and Alexandre Gramfort. Non-linear auto-regressive models for cross-frequency coupling in neural time series. *PLoS computational biology*, 13(12):e1005893, 2017.
- [91] Haiteng Jiang, Ali Bahramisharif, Marcel AJ van Gerven, and Ole Jensen. Measuring

- directionality between neuronal oscillations of different frequencies. *Neuroimage*, 118:359–367, 2015.
- [92] Lionel Barnett, Adam B Barrett, and Anil K Seth. Granger causality and transfer entropy are equivalent for gaussian variables. *Physical review letters*, 103(23):238701, 2009.
 - [93] Joseph T Lizier. Measuring the dynamics of information processing on a local scale in time and space. In *Directed Information Measures in Neuroscience*, pages 161–193. Springer, 2014.
 - [94] György Buzsáki, Caroline Geisler, Darrell A Henze, and Xiao-Jing Wang. Interneuron diversity series: circuit complexity and axon wiring economy of cortical interneurons. *Trends in neurosciences*, 27(4):186–193, 2004.
 - [95] Diego Lozano-Soldevilla, Niels ter Huurne, and Robert Oostenveld. Neuronal oscillations with non-sinusoidal morphology produce spurious phase-to-amplitude coupling and directionality. *Frontiers in computational neuroscience*, 10:87, 2016.
 - [96] Jon López-Azcárate, Mikel Tainta, María C Rodríguez-Oroz, Miguel Valencia, Rafael González, Jorge Guridi, Jorge Iriarte, José A Obeso, Julio Artieda, and Manuel Alegre. Coupling between beta and high-frequency activity in the human subthalamic nucleus may be a pathophysiological mechanism in parkinson’s disease. *Journal of Neuroscience*, 30(19):6667–6677, 2010.
 - [97] Spase Petkoski and Viktor K Jirsa. Transmission time delays organize the brain network synchronization. *Philosophical Transactions of the Royal Society A*, 377(2153):20180132, 2019.
 - [98] Paul A Felts, Theresa A Baker, and Kenneth J Smith. Conduction in segmentally demyelinated mammalian central axons. *Journal of Neuroscience*, 17(19):7267–7277, 1997.
 - [99] Thomas J Whitford, Judith M Ford, Daniel H Mathalon, Marek Kubicki, and Martha E Shenton. Schizophrenia, myelination, and delayed corollary discharges: a hypothesis. *Schizophrenia bulletin*, 38(3):486–494, 2012.
 - [100] Joseph T Lizier. *The local information dynamics of distributed computation in complex systems*. Springer Science & Business Media, 2012.
 - [101] Norbert Wiener. The theory of prediction. *Modern mathematics for engineers*, 1956.
 - [102] Terry Bossomaier, Lionel Barnett, Michael Harré, and Joseph T Lizier. An introduction to transfer entropy. *Cham: Springer International Publishing*, pages 65–95,

2016.

- [103] Joseph T Lizier and Mikhail Prokopenko. Differentiating information transfer and causal effect. *The European Physical Journal B*, 73(4):605–615, 2010.
- [104] Joseph T Lizier, Mikhail Prokopenko, and Albert Y Zomaya. Local information transfer as a spatiotemporal filter for complex systems. *Physical Review E*, 77(2):026110, 2008.
- [105] András Antos and Ioannis Kontoyiannis. Convergence properties of functional estimates for discrete distributions. *Random Structures & Algorithms*, 19(3-4):163–193, 2001.
- [106] Young-Il Moon, Balaji Rajagopalan, and Upmanu Lall. Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3):2318, 1995.
- [107] Ralf Steuer, Jürgen Kurths, Carsten O Daub, Janko Weise, and Joachim Selbig. The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics*, 18(suppl_2):S231–S240, 2002.
- [108] Roland L Dobrushin. Prescribing a system of random variables by conditional distributions. *Theory of Probability & Its Applications*, 15(3):458–486, 1970.
- [109] Oldrich Vasicek. A test for normality based on sample entropy. *Journal of the Royal Statistical Society: Series B (Methodological)*, 38(1):54–59, 1976.
- [110] Joseph T Lizier. Jidt: An information-theoretic toolkit for studying the dynamics of complex systems. *Frontiers in Robotics and AI*, 1:11, 2014.
- [111] Liangyue Cao. Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D: Nonlinear Phenomena*, 110(1-2):43–50, 1997.
- [112] Mario Ragwitz and Holger Kantz. Markov models from data by simple nonlinear time series predictors in delay embedding spaces. *Physical Review E*, 65(5):056201, 2002.
- [113] Ramon Martínez-Cancino, Arno Delorme, Kreutz-Delgado Kenneth, and Makeig Scott. Computing phase amplitude coupling in eeglab: Pactools. *Proceedings of the 20th International IEEE Conference on BioInformatics And BioEngineering*, 2020.
- [114] Tom Dupre La Tour. *Nonlinear models for neurophysiological time series*. PhD thesis, 2018.
- [115] jidt/readme-schreibertedemos.pdf at master · jlizier/jidt. <https://github.com/>

jlizier/jidt/blob/master/demos/octave/SchreiberTransferEntropyExamples/
 README-SchreiberTeDemos.pdf. (Accessed on 09/11/2020).

- [116] Julie A Onton and Scott Makeig. High-frequency broadband modulation of electroencephalographic spectra. *Frontiers in human neuroscience*, 3:61, 2009.
- [117] Ryan T. Canolty and Robert T. Knight. The functional role of cross-frequency coupling. *Trends in Cognitive Sciences*, 14, November 2010.
- [118] Tao Zhang, Xiaxia Xu, and Chenguang Zheng. Reduction in lfp cross-frequency coupling between theta and gamma rhythms associated with impaired stp and ltp in a rat model of brain ischemia. *Frontiers in Computational Neuroscience*, 7:27, 2013.
- [119] Juergen Fell and Nikolai Axmacher. The role of phase synchronization in memory processes. *Nature reviews neuroscience*, 12(2):105–118, 2011.
- [120] Björn Rasch and Jan Born. About sleep’s role in memory. *Physiological reviews*, 2013.
- [121] Til O Bergmann and Jan Born. Phase-amplitude coupling: a general mechanism for memory processing and synaptic plasticity? *Neuron*, 97(1):10–13, 2018.
- [122] Charles E Schroeder and Peter Lakatos. The gamma oscillation: master or slave? *Brain topography*, 22(1):24–26, 2009.
- [123] Charles E Schroeder and Peter Lakatos. Low-frequency neuronal oscillations as instruments of sensory selection. *Trends in neurosciences*, 32(1):9–18, 2009.
- [124] Roy Mukamel, Hagar Gelbard, Amos Arieli, Uri Hasson, Itzhak Fried, and Rafael Malach. Coupling between neuronal firing, field potentials, and fmri in human auditory cortex. *Science*, 309(5736):951–954, 2005.
- [125] Jörn Niessing, Boris Ebisch, Kerstin E Schmidt, Michael Niessing, Wolf Singer, and Ralf AW Galuske. Hemodynamic signals correlate tightly with synchronized gamma oscillations. *science*, 309(5736):948–951, 2005.
- [126] Johanna Wagner, Scott Makeig, David Hoopes, and Mateusz Gola. Can oscillatory alpha-gamma phase-amplitude coupling be used to understand and enhance tms effects? *Frontiers in human neuroscience*, 13:263, 2019.
- [127] Leonardo Novelli, Patricia Wollstadt, Pedro Mediano, Michael Wibral, and Joseph T Lizier. Large-scale directed network inference with multivariate transfer entropy and hierarchical statistical testing. *Network Neuroscience*, 3(3):827–847, 2019.

- [128] Michael Lindner, Raul Vicente, Viola Priesemann, and Michael Wibral. Trentool: A matlab open source toolbox to analyse information flow in time series data with transfer entropy. *BMC neuroscience*, 12(1):119, 2011.
- [129] M Chavez, M Besserve, C Adam, and J Martinerie. Towards a proper estimation of phase synchronization from time series. *Journal of Neuroscience methods*, 154(1-2):149–160, 2006.
- [130] Dino Dvorak and André A Fenton. Toward a proper estimation of phase–amplitude coupling in neural oscillations. *Journal of Neuroscience methods*, 225:42–56, 2014.
- [131] Alexandre Hyafil. Misidentifications of specific forms of cross-frequency coupling: three warnings. *Frontiers in neuroscience*, 9:370, 2015.
- [132] Immo Weber, Esther Florin, Michael Von Papen, and Lars Timmermann. The influence of filtering and downsampling on the estimation of transfer entropy. *Plos one*, 12(11):e0188210, 2017.
- [133] Patricia Wollstadt, Kristin K Sellers, Lucas Rudelt, Viola Priesemann, Axel Hutt, Flavio Fröhlich, and Michael Wibral. Breakdown of local information processing may underlie isoflurane anesthesia effects. *PLoS computational biology*, 13(6):e1005511, 2017.
- [134] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.
- [135] Esther Florin, Joachim Gross, Johannes Pfeifer, Gereon R Fink, and Lars Timmermann. The effect of filtering on granger causality based multivariate causality measures. *Neuroimage*, 50(2):577–588, 2010.
- [136] Lionel Barnett and Anil K Seth. Behaviour of granger causality under filtering: theoretical invariance and practical application. *Journal of neuroscience methods*, 201(2):404–419, 2011.
- [137] Edoardo Pinzuti, Patricia Wollstadt, Aaron Gutknecht, Oliver Tüscher, and Michael Wibral. Measuring spectrally-resolved information transfer for sender-and receiver-specific frequencies. *bioRxiv*, 2020.
- [138] Luca Faes, Giandomenico Nollo, and Alberto Porta. Information-based detection of nonlinear granger causality in multivariate processes via a nonuniform embedding technique. *Physical Review E*, 83(5):051112, 2011.
- [139] Joseph Lizier and Mikail Rubinov. Multivariate construction of effective computa-

tional networks from observational data. 2012.

- [140] Patricia Wollstadt, Joseph T Lizier, Raul Vicente, Conor Finn, Mario Martinez-Zarzuela, Pedro Mediano, Leonardo Novelli, and Michael Wibral. Idtxl: The information dynamics toolkit xl: a python package for the efficient analysis of multivariate information dynamics in networks. *arXiv preprint arXiv:1807.10459*, 2018.
- [141] Etienne Combrisson, Timothy Nest, Andrea Brovelli, Robin Ince, Juan Soto, Aymeric Guillot, and Karim Jerbi. Tensorpac: an open-source python toolbox for tensor-based phase-amplitude coupling measurement in electrophysiological brain signals. 2020.
- [142] Christian Kothe. *fast-serialize-deserialize*, 2020.
- [143] Daniel G Wakeman and Richard N Henson. A multi-subject, multi-modal human neuroimaging dataset. *Scientific data*, 2:150001, 2015.
- [144] Ramon Martinez-Cancino, Arnaud Delorme, and Scott Makeig. Analyzing eeg data in eeglab: One useful approach. *SfN*, 2018, 2018.
- [145] Scott Makeig, Ramon Martinez-Cancino, Makoto Miyakoshi, Zeynep Akalin Acar, Luca Pion-Tonachini, John Iversen, Cyril Pernet, and Arnaud Delorme. Assessing event-related eeg brain dynamics using eeglab. *In preparation*, 2020.
- [146] Mattia Pagnotta. *The Dynamic Reorganization of Cortical Network Interactions during Visual Selective Attention*. PhD thesis, Université de Fribourg, 2020.
- [147] Mattia F Pagnotta, David Pascucci, and Gijs Plomp. Nested oscillations and brain connectivity during sequential stages of feature-based attention. *bioRxiv*, 2020.
- [148] Arnaud Delorme, Amitava Majumdar, Subhashini Sivagnanam, Ramon Martinez-Cancino, Kenneth Yoshimoto, and Scott Makeig. The open eeglab portal. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1142–1145. IEEE, 2019.
- [149] Lindsay M Alexander, Jasmine Escalera, Lei Ai, Charissa Andreotti, Karina Febre, Alex Mangone, Natan Vega Potler, Nicolas Langer, Alexis Alexander, Meagan Kovacs, et al. The healthy brain network biobank: An open resource for transdiagnostic research in pediatric mental health and learning disorders. 2017.
- [150] David O’Connor, Natan Vega Potler, Meagan Kovacs, Ting Xu, Lei Ai, John Pellman, Tamara Vanderwal, Lucas C Parra, Samantha Cohen, Satrajit Ghosh, et al. The healthy brain network serial scanning initiative: a resource for evaluating inter-individual differences and their reliabilities across scan conditions and sessions.

Gigascience, 6(2):giw011, 2017.

- [151] Andreas Pedroni, Amirreza Bahreini, and Nicolas Langer. Automagic: Standardized preprocessing of big eeg data. *Neuroimage*, 200:460–473, 2019.
- [152] G Schumann, E Loth, T Banaschewski, A Barbot, G Barker, C Büchel, PJ Conrod, JW Dalley, H Flor, J Gallinat, et al. The imagen study: reinforcement-related behaviour in normal brain function and psychopathology. *Molecular psychiatry*, 15(12):1128–1139, 2010.
- [153] Cathie Sudlow, John Gallacher, Naomi Allen, Valerie Beral, Paul Burton, John Danesh, Paul Downey, Paul Elliott, Jane Green, Martin Landray, et al. Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *Plos med*, 12(3):e1001779, 2015.
- [154] David C Van Essen, Kamil Ugurbil, Edward Auerbach, Deanna Barch, TEJ Behrens, Richard Bucholz, Acer Chang, Liyong Chen, Maurizio Corbetta, Sandra W Curtiss, et al. The human connectome project: a data acquisition perspective. *Neuroimage*, 62(4):2222–2231, 2012.
- [155] Adriana Di Martino, Chao-Gan Yan, Qingyang Li, Erin Denio, Francisco X Castellanos, Kaat Alaerts, Jeffrey S Anderson, Michal Assaf, Susan Y Bookheimer, Mirella Dapretto, et al. The autism brain imaging data exchange: towards a large-scale evaluation of the intrinsic brain architecture in autism. *Molecular psychiatry*, 19(6):659–667, 2014.
- [156] Susanne G Mueller, Michael W Weiner, Leon J Thal, Ronald C Petersen, Clifford R Jack, William Jagust, John Q Trojanowski, Arthur W Toga, and Laurel Beckett. Ways toward an early diagnosis in alzheimer’s disease: the alzheimer’s disease neuroimaging initiative (adni). *Alzheimer’s & Dementia*, 1(1):55–66, 2005.
- [157] Nima Bigdely-Shamlo, Jeremy Cockfield, Scott Makeig, Thomas Rognon, Chris La Valle, Makoto Miyakoshi, and Kay A Robbins. Hierarchical event descriptors (hed): semi-structured tagging for real-world events in large-scale eeg. *Frontiers in neuroinformatics*, 10:42, 2016.
- [158] Nima Bigdely-Shamlo, Tim Mullen, Christian Kothe, Kyung-Min Su, and Kay A Robbins. The prep pipeline: standardized preprocessing for large-scale eeg analysis. *Frontiers in neuroinformatics*, 9:16, 2015.
- [159] Laurel J Gabard-Durnam, Adriana S Mendez Leal, Carol L Wilkinson, and April R Levin. The harvard automated processing pipeline for electroencephalography (happe): standardized processing software for developmental and high-artifact data. *Frontiers in neuroscience*, 12:97, 2018.

- [160] Krzysztof J Gorgolewski, Tibor Auer, Vince D Calhoun, R Cameron Craddock, Samir Das, Eugene P Duff, Guillaume Flandin, Satrajit S Ghosh, Tristan Glatard, Yaroslav O Halchenko, et al. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific data*, 3(1):1–9, 2016.
- [161] Guiomar Niso, Krzysztof J Gorgolewski, Elizabeth Bock, Teon L Brooks, Guillaume Flandin, Alexandre Gramfort, Richard N Henson, Mainak Jas, Vladimir Litvak, Jeremy T Moreau, et al. Meg-bids, the brain imaging data structure extended to magnetoencephalography. *Scientific data*, 5(1):1–5, 2018.
- [162] Christopher Holdgraf, Stefan Appelhoff, Stephan Bickel, Kristofer Bouchard, Sasha D’Ambrosio, Olivier David, Orrin Devinsky, Benjamin Dichter, Adeen Flinker, Brett L Foster, et al. ieeg-bids, extending the brain imaging data structure specification to human intracranial electrophysiology. *Scientific data*, 6(1):1–6, 2019.
- [163] Cyril R Pernet, Stefan Appelhoff, Krzysztof J Gorgolewski, Guillaume Flandin, Christophe Phillips, Arnaud Delorme, and Robert Oostenveld. Eeg-bids, an extension to the brain imaging data structure for electroencephalography. *Scientific Data*, 6(1):1–5, 2019.
- [164] Zeynep Akalin Acar and Scott Makeig. Neuroelectromagnetic forward head modeling toolbox. *Journal of neuroscience methods*, 190(2):258–270, 2010.
- [165] Denis Brunet, Micah M Murray, and Christoph M Michel. Spatiotemporal analysis of multichannel eeg: Cartool. *Computational intelligence and neuroscience*, 2011, 2011.
- [166] Carsten Hermann Wolters, M Kuhn, Alfred Anwander, and S Reitzinger. A parallel algebraic multigrid solver for finite element method based source localization in the human brain. *Computing and visualization in science*, 5(3):165–177, 2002.
- [167] Scott Makeig, Marissa Westerfield, T-P Jung, Sonia Enghoff, Jeanne Townsend, Eric Courchesne, and Terrence J Sejnowski. Dynamic brain sources of visual evoked responses. *Science*, 295(5555):690–694, 2002.
- [168] Julie Onton, Arnaud Delorme, and Scott Makeig. Frontal midline eeg dynamics during working memory. *Neuroimage*, 27(2):341–356, 2005.
- [169] Gert Pfurtscheller and FH Lopes Da Silva. Event-related eeg/meg synchronization and desynchronization: basic principles. *Clinical neurophysiology*, 110(11):1842–1857, 1999.
- [170] Fiorenzo Artoni, Danilo Menicucci, Arnaud Delorme, Scott Makeig, and Silvestro

- Micera. Relica: a method for estimating the reliability of independent components. *Neuroimage*, 103:391–400, 2014.
- [171] Scott Makeig, Anthony J Bell, Tzyy-Ping Jung, Terrence J Sejnowski, et al. Independent component analysis of electroencephalographic data. *Advances in neural information processing systems*, pages 145–151, 1996.
 - [172] Fabien Lotte, Laurent Bougrain, Andrzej Cichocki, Maureen Clerc, Marco Congedo, Alain Rakotomamonjy, and Florian Yger. A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update. *Journal of neural engineering*, 15(3):031005, 2018.
 - [173] Scott Makeig, Christian Kothe, Tim Mullen, Nima Bigdely-Shamlo, Zhilin Zhang, and Kenneth Kreutz-Delgado. Evolving signal processing for brain–computer interfaces. *Proceedings of the IEEE*, 100(Special Centennial Issue):1567–1584, 2012.
 - [174] Nancy Wilkins-Diehr, Dennis Gannon, Gerhard Klimeck, Scott Oster, and Sudhakar Pamidighantam. Teragrid science gateways and their impact on science. *Computer*, 41(11):32–41, 2008.
 - [175] Subhashini Sivagnanam, Amit Majumdar, Kenneth Yoshimoto, Vadim Astakhov, Anita E Bandrowski, Maryann E Martone, and Nicholas T Carnevale. Introducing the neuroscience gateway. *IWSG*, 993, 2013.
 - [176] Subhashini Sivagnanam, Kenneth Yoshimoto, Nicholas T Carnevale, and Amit Majumdar. The neuroscience gateway: Enabling large scale modeling and data processing in neuroscience. In *Proceedings of the Practice and Experience on Advanced Research Computing*, pages 1–7. 2018.
 - [177] Michael L Hines and N Ted Carnevale. Translating network models to parallel hardware in neuron. *Journal of neuroscience methods*, 169(2):425, 2008.
 - [178] James M Bower and David Beeman. *The book of GENESIS: exploring realistic neural models with the GEneral NEural SIMulation System*. Springer Science & Business Media, 2012.
 - [179] Subhasis Ray, Raamesh Deshpande, Niraj Dudani, and Upinder S Bhalla. A general biological simulator: the multiscale object oriented simulation environment, moose. *BMC Neuroscience*, 9(1):1–2, 2008.
 - [180] Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.
 - [181] Dan FM Goodman and Romain Brette. The brian simulator. *Frontiers in neuro-*

science, 3:26, 2009.

- [182] Andrew P Davison, Daniel Brüderle, Jochen M Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. Pynn: a common interface for neuronal network simulators. *Frontiers in neuroinformatics*, 2:11, 2009.
- [183] Bruce Fischl. Freesurfer. *Neuroimage*, 62(2):774–781, 2012.
- [184] J-Donald Tournier, Fernando Calamante, and Alan Connolly. Mrtrix: diffusion tractography in crossing fiber regions. *International journal of imaging systems and technology*, 22(1):53–66, 2012.
- [185] Mark A Miller, Wayne Pfeiffer, and Terri Schwartz. Creating the cipres science gateway for inference of large phylogenetic trees. In *2010 gateway computing environments workshop (GCE)*, pages 1–8. Ieee, 2010.
- [186] Mark A Miller, Wayne Pfeiffer, and Terri Schwartz. The cipres science gateway: a community resource for phylogenetic analyses. In *Proceedings of the 2011 TeraGrid Conference: extreme digital discovery*, pages 1–8, 2011.
- [187] Bhanu Rekapalli, Paul Giblock, and Christopher Reardon. Poplar: portal for petascale lifescience applications and research. *BMC bioinformatics*, 14(9):1–12, 2013.
- [188] Subhashini Sivagnanam, Amit Majumdar, Kenneth Yoshimoto, Vadim Astakhov, Anita Bandrowski, MaryAnn Martone, and Nicholas T Carnevale. Early experiences in developing and managing the neuroscience gateway. *Concurrency and Computation: Practice and Experience*, 27(2):473–488, 2015.
- [189] Mark A Miller, Terri Schwartz, Brett E Pickett, Sherry He, Edward B Klem, Richard H Scheuermann, Maria Passarotti, Seth Kaufman, and Maureen A O’Leary. A restful api for access to phylogenetic tools via the cipres science gateway. *Evolutionary Bioinformatics*, 11:EBO–S21501, 2015.
- [190] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.
- [191] Te-Won Lee, Mark Girolami, and Terrence J Sejnowski. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural computation*, 11(2):417–441, 1999.